NEURAL AND ANALOG COMPUTATION ON RECONFIGURABLE MIXED-SIGNAL PLATFORMS

A Thesis
Presented to
The Academic Faculty

by

Stephen Nease

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2014

NEURAL AND ANALOG COMPUTATION ON RECONFIGURABLE MIXED-SIGNAL PLATFORMS

Approved by:

Maysam Ghovanloo, Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Jennifer Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Aaron Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology

Hua Wang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Bradley Minch
Department of Electrical and Computer Engineering
Franklin W. Olin College of Engineering

Date Approved: May 15 2014

For Mom, Dad, and Will

ACKNOWLEDGEMENTS

This dissertation is a product of the work, advice, and love of many people. I hope I can give them their due in this short space.

First and foremost, there is no question in my mind that this dissertation would not have been completed without the love and support of my family. My father's intellectual curiosity was my inspiration for venturing into academia. I will do my best to model his humility and kindness towards others as I make my next professional steps. My mother's constant support has been a blessing over the past five years. Her unwavering confidence that I could complete this work has boosted my spirits when I needed it most. My brother's enthusiasm about my work has kept me striving to live up to his view of it, and his wonderful way with people is a trait that I attempt to emulate every day. My aunts, uncles, and cousins have also shown a great interest in my work, which kept me motivated throughout the program. Thanks to all of you.

I owe a debt of gratitude to Rose-Hulman and the academic mentors I was fortunate to meet there. Dr. Tina Hudson was the first person to tell me that I was capable of completing the degree, and her help in choosing a specialty and school was invaluable. Her passion for engineering is infectious, and her energetic teaching style is something that I have attempted to emulate in my own lectures. Dr. Ed Doering gave me my first engineering job, taught me fundamentals of digital logic, and introduced me to electronic music synthesis. Chapter 5 of this dissertation is a direct result of taking his course on the subject. Dr. Wayne Padgett gave me solid fundamentals in signal processing, but more importantly, he taught me the importance of finding and valuing good academic texts. Finally, Dr. Mario Simoni spent two quarters teaching me the fundamentals of VLSI design. I appreciated his willingness to spend a great deal of time with me – even teaching me aVLSI one-on-one for half a quarter. These four educators are just some examples of the wonderful teachers I met at Rose-Hulman, and I thank them all.

I also thank my research adviser, Dr. Jennifer Hasler. Her research interests align with mine quite closely, and it has been a joy to explore what I am interested in over the past five years. Her knowledge of circuits, device physics, and neuroscience is both wide-ranging and deep, and I have enjoyed learning those disciplines from her. She gave me a number of opportunities which changed my perspective on academia. In particular, co-teaching the neuromorphic engineering class was a very challenging and rewarding experience, and it showed me that I enjoy that aspect of academia. Our trip to Capocaccia allowed me to interact with other researchers and make connections in academia that I hope will last for years. I am very grateful to Jen for all the opportunities she gave me.

Finally, I would like to thank all my labmates, past and present. Their daily support and help has sustained me through the trials of the grad school. I would like to mention a few specific people who have been instrumental to my success. For the first few years of the program, I looked to Stephen Brink as an exemplary student, researcher, and friend. Not only is his scientific curiosity and knowledge enviable, but he is also one of the most selfless people I've ever met. Working on the Neuron1D chip with him led to the RASP 3.0H chip, a major contribution of my dissertation. Suma George has been a constant source of support over the years, and I am thankful to her for her positive attitude and willingness to work through technical problems with me. Richie Wunderlich's incredible creativity when solving problems has been an inspiration. His intuitive understanding of circuits and physics has pushed me to improve my own understanding, and I have valued his friendship greatly. Farhan Adil's experience and circuit intuition has been an enormous help to me. I greatly enjoyed collaborating with him towards the end of our grad school careers. I would also like to mention my former labmates Shubha, Craig, Scott, Sam, and Arindam. Each one taught me a great deal, and I value both their professional expertise and their friendship. I also value the hard work of other current labmates – Michelle, Sihwan, and Andrew. I wish you the best over the next few years, and I hope grad school rewards you as it has me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This work addresses neural and analog computation on reconfigurable mixed-signal platforms. Many engineered systems could gain tremendous benefits by emulating neural systems. For example, neural systems are incredibly power efficient and fault-tolerant. They are also capable of types of computation that we cannot yet match with conventional computers. Neuromorphic engineers typically implement neural computation using analog circuits because they are low-power and naturally model some aspects of neurobiology. One problem with analog circuits is that they are typically inflexible. To address this shortcoming, our lab has developed reconfigurable analog systems known as Field Programmable Analog Arrays (FPAAs).

This dissertation consists of two main parts. The first is the implementation of neural and analog circuits on FPAAs. We first implemented an adaptive winner-take-all circuit, which could model attention in neural systems. Next, we modeled the dendrite, which is the conductive tissue that relays inputs from synapses to the neuron cell body. We also implemented a subtractive music synthesizer, perhaps providing the electronic music synthesis community with a good platform for experimentation. Finally, we conducted a number of neural learning experiments on a neuromorphic platform.

The second part of this dissertation includes design aspects of new FPAAs, including configurable blocks that can be used as current-mode DACs in a digitally-enhanced FPAA, the RASP 2.9v. We also consider the design of a new neuromorphic platform containing 256 neurons and over 200,000 synapses, many with learning capability. We also created an active delay line that could be used for beamforming or FIR filter applications.

In summary, this work adds to the field of reconfigurable systems by both showing how to implement circuits with them and creating new systems based on lessons learned while working with previous systems.

CHAPTER I

INTRODUCTION

This dissertation details novel contributions to neuromorphic engineering and analog signal processing. The majority of these contributions are in the context of Field-Programmable Analog Arrays (FPAAs), which are Very Large Scale Integrated (VLSI) circuits that can be programmed to implement many different analog subcircuits. The term "FPAA" is quickly becoming a misnomer, since the most recent generation of FPAAs includes both digital and analog circuits – hence the term "Mixed-Signal" in the title of this dissertation. This work consists, broadly, of two different types of tasks. The first is the implementation of low-power analog signal processing systems on these FPAAs; we took existing hardware systems and compiled and tested new subcircuits on them. This informed the second type of task, namely the design and layout of new FPAAs. Based on our testing of the older hardware, we added enhancements to the FPAAs, which increased their signal processing capabilities.

This chapter briefly outlines what neuromorphic and analog signal processing are, and why we care about studying them. It introduces the basic concepts of FPAAs and the key technology that enables their reconfigurability – floating-gate transistors. Finally, the rest of the dissertation is outlined.

1.1 What is Neuromorphic Engineering, and Why Study it?

Neuromorphic engineering is a subfield of electrical engineering that attempts to build hardware that efficiently models various aspects of neurobiological systems observed in nature. Sometimes the modifier "neuromorphic" is used in other fields to describe biologically-inspired projects in that field (neuromorphic algorithms in computer science, for example), but the original definition of neuromorphic engineering was as a subset of electrical engineering. The founder of neuromorphic engineering is Carver Mead, whose groundbreaking book on the subject launched an entire field [44] in the late 1980s.

There are three main attributes of the brain that make it valuable to emulate: its algorithmic capabilities, its fault-tolerant architecture, and its power efficiency. The human brain is an incredible machine that is capable of performing algorithmic tasks that have not yet been matched by any single engineered system. The nervous system's computational capabilities are obvious to anyone who considers all the functions that our brains perform in a day – speech recognition, motor control, object detection, logical reasoning, learning, and much more. One question that might arise is "don't the fields of artificial intelligence and machine learning already seek to make systems that do these tasks?" The answer is yes, but there is an important caveat – artificial intelligence and machine learning do not require that the hardware on which they are run be biologically inspired. This has important consequences in terms of the efficiency of the final systems.

The brain is incredibly efficient; all of the tasks described above are run on a power budget of about 20W. A primary reason to study neuromorphic engineering is that bio-inspired hardware is much more power-efficient than traditional computer architectures. Much neuromorphic engineering is done with analog circuits, which are inherently more power efficient at low signal-to-noise ratios (SNRs) [50].

Neuromorphic engineers claim that transistors used in an analog sense can be used to emulate biological processes. Neuromorphic engineering often begins with the principle that the transistor acts as a biological analog. Carver Mead recognized that both both silicon and biological channels behave according to the same natural principle. The channel of a transistor operated in its subthreshold regime is governed by the diffusion equation, as are many biological processes [44].

The channel of a transistor is a region of silicon that separates the drain from the source (see Fig. 1a). This area forms an energy barrier to charge carriers at the source and at the drain. The number of charge carriers at the source or drain end of the channel is determined by the size of this barrier, which is modulated by the difference between the gate voltage and the source or drain voltage. Since the source is operated at a higher potential than the drain in the P-channel device, the barrier at the source end of the channel is lower, so there are more charge carriers at the source end of the channel than at the drain end. Therefore,

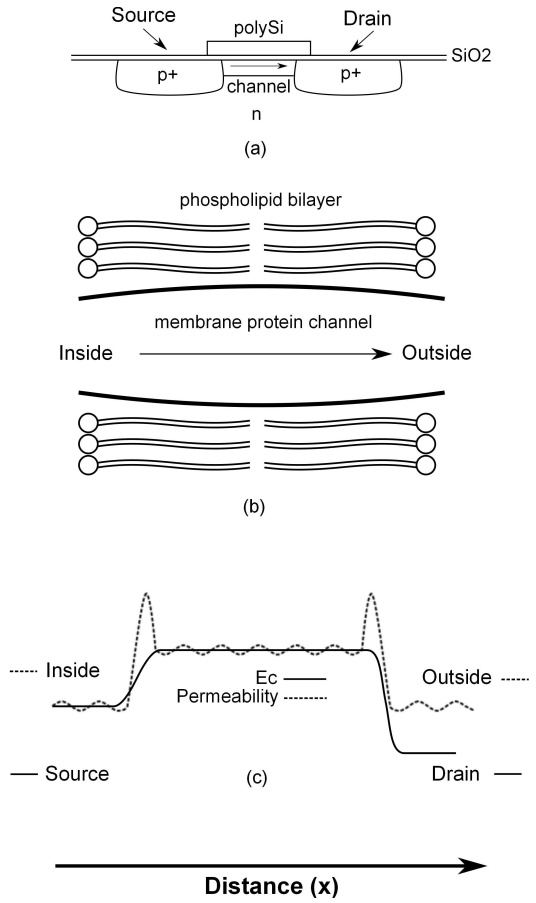


Figure 1: (a) The physical structure of a MOSFET consists of polysilicon, silicon dioxide, and doped n-type silicon. A channel is formed between the source and the drain. (b) The physical structure of a biological channel consists of an insulating phospholipid bilayer and a protein which stretches across the barrier. The protein is the channel in this case. (c) The band diagram of silicon (solid line) has a similar shape to the classical model of membrane permeability proposed by Danielli [28] (dashed line). In both cases, carriers must overcome energy barriers to travel from one side of the device to the other.

we have a gradient of charge carriers from the source end of the channel to the drain end. This is illustrated in Fig. 1c. This means that carriers must diffuse from the source to the drain according to the diffusion equation from [44]:

$$v_{diffusion} = -D\frac{1}{N}\frac{dN}{dh}, \tag{1}$$

where $v_{diffusion}$ is the velocity of carriers, D is the diffusion constant, N is the number of charge carriers per unit volume, and h is distance. When the diffusion equation is applied in the case of a gradient of charge carriers from the source to the drain of a pFET channel,

the current is given in [37] as

$$\begin{aligned} I &= I_0 e^{\kappa(V_{dd}-V_g)/U_T}\left(e^{-(V_{dd}-V_s)/U_T} - e^{-(V_{dd}-V_d/U_T)}\right) \\ &= I_0' e^{-\kappa V_g/U_T}\left(e^{V_s/U_T} - e^{V_d/U_T}\right). \end{aligned} \tag{2}$$

V_{dd} is the well potential of the pFET, V_g is the gate voltage, V_s is the source voltage, and V_d is the drain voltage, all referenced to ground. I_0 is a collection of physical constants determined by the device's layout and the technology in which it is implemented. κ is a measure of how well the gate voltage modulates the potential at the channel's surface. U_T is the thermal voltage (typically around 26 mV at room temperature). To simplify the nomenclature, we can reference the terminal voltages to V_{dd}, in which case $I_0' = I_0$. To reference everything to ground, we let $I_0' = I_0 e^{\kappa V_{dd}/U_T} e^{-V_{dd}/U_T}$.

The idea of overcoming energy barriers to produce current is also seen in biological channels. Fig. 1b, illustrates the structure of a channel embedded in a membrane. Fig. 1c shows how both biological and silicon channels generate barriers to current, where the barrier is shown as a change in membrane permeability in the case of biological channels and a change in potential energy in the case of silicon channels.

The last reason one would want to emulate neural systems is that they are insensitive to faults in their fundamental building blocks. The brain's architecture is highly parallel, which has the consequence that the failure of any particular neuron or synapse will not affect the operation of the entire system. Synaptic transmission is quite unreliable – its failure rate is over 50% [1]. In contrast to the brain, a digital system that has just one transistor fail in a critical portion of the system could destroy its operation.

In summary, it is important to study neuromorphic engineering because it offers engineers the promise of highly capable, low-power, and fault-tolerant computational machines. The next section addresses the key ingredient to achieving low power in neuromorphic systems – analog signal processing.

1.2 Analog Signal Processing and the Importance of Reconfigurability

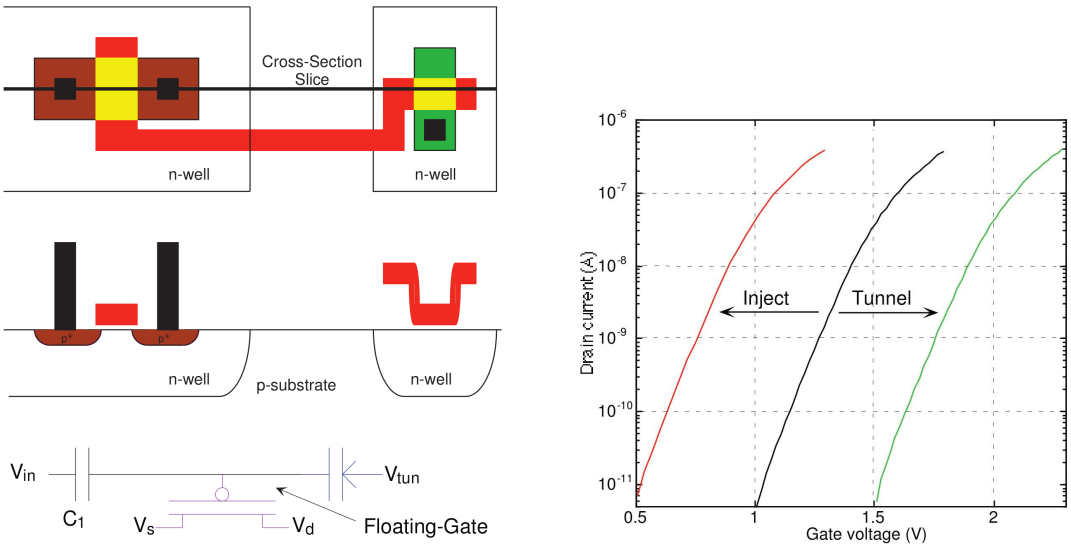
Analog signals are continuous-valued signals in continuous time. Typically they are encoded using a voltage or current, and data processing is performed by passing signals through analog primitives, such as capacitors, inductors, resistors, common-source/gate/drain amplifiers, operational transconductance amplifiers (OTAs), Multiple Input Translinear Elements (MITEs), etc. Analog signal processing is inherently more power efficient than digital signal processing at moderate SNRs. But digital signal processing currently dominates the marketplace.

One of the reasons analog circuits have not been utilized as much as they could be is due to their relative inflexibility. Once a traditional analog system has been designed, there is no way to significantly modify its architecture. One must fabricate the analog design, test it, and then re-fabricate if there are problems with the design's architecture. While one can use circuit simulation tools to virtually test a subcircuit, it is not feasible to simulate incredibly large analog signal processing systems. The capability to digitally trim analog systems improves their flexibility, but this is not as flexible as the ability to completely change architectures – a feature of digital Fiesd-Programmable Analog Arrays (FPGAs). So it is desireable to create analog systems that can be changed at an architectural level for rapid prototyping and deployment,enabling a quick turnaround in the creation of low-power analog signal processing systems.

Our lab has produced such systems, known as Field-Programmable Analog Arrays (FPAAs). A key technology enabling them is the floating-gate transistor. The next section discusses the floating-gate transistor, and the following section introduces FPAAs.

1.3 The Floating-Gate Transistor

Reconfigurable analog systems are enabled by a key piece of technology – the floating-gate transistor. A floating-gate (FG) FET is much like a normal transistor, except its gate has no DC path to ground. Voltage is applied to the gate purely capacitively. An image of the layout of a pFET floating-gate transistor is shown in Fig. 2(a). The lack of a DC path to ground means that once charge is stored on the gate, it will remain there without the



(a) (b)

Figure 2: (a) Layout of a floating-gate transistor; image from [22]. (b) Injection and tunneling modify the floating-gate voltage and thereby change the current-voltage characteristics; image from [26].

need for a directly-applied potential. We are able to place charge on the gate and remove charge from it using the physical processes of Fowler-Nordheim tunneling and hot electron injection. This changes the current-voltage characteristics of the FET, as shown in Fig. 2(b).

We have already introduced an equation describing the subthreshold behavior of the pFET in all regions of operation. When the drain to source potential is greater than approximately $4U_T$, the transistor is said to operate in its saturation regime. The modeling equation we use to design with floating-gate pFETs in the subthreshold saturation regime is

$$I = I_0 e^{\left(\kappa(V_{dd}-V_{fg})-(V_{dd}-V_s)+\sigma(V_{dd}-V_d)\right)/U_T}, \tag{3}$$

where κ is the capacitive divider between the floating-gate node and the transistor's surface potential, V_{fg} is the potential at the floating gate, V_{dd} is the bulk potential, σ models drain coupling, and U_T is the thermal voltage. The floating-gate voltage V_{fg} is determined by summing the contributions of capacitively-coupled voltages into the node, as well as the initial charge at the node Q_0 (which is set by tunneling and injection):

$$V_{fg} = \frac{Q_0}{C_T} + \sum_{i=1}^{n} V_i \frac{C_i}{C_T}, \tag{4}$$

where C_T is the total capacitance of the floating node, V_i is a potential applied to the node through a capacitor, and C_i is the value of the capacitor between V_i and the floating node.

Reconfigurability in FPAAs is achieved with FGs in two ways. First, reconfigurability on the circuit scale is achieved by modifying the floating-gate charge to trim analog subcircuits. For example, changing the bias current in an OTA-C low pass filter will change the cutoff frequency of the filter. Second, architecture reconfigurability is achieved by treating FGs as on/off switches and using them to route between subsystems of an architecture. This is the idea behind an FPAA, which is discussed in the next section.

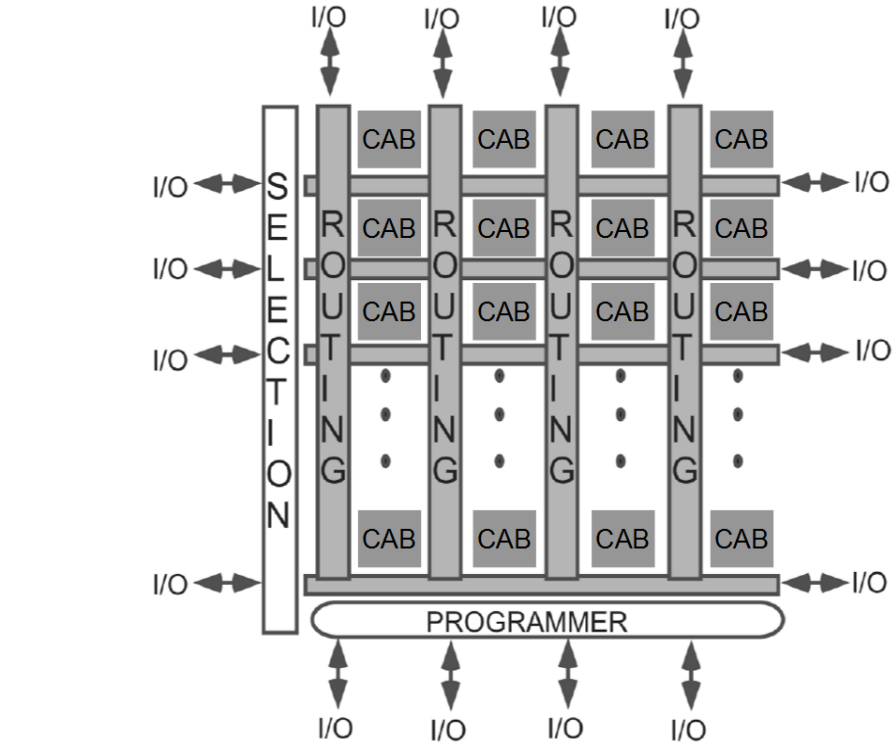
1.4 Introduction to Field-Programmable Analog Arrays

The RASP 2.8a is one in a family of Field-Programmable Analog Arrays (FPAAs). An FPAA is a mixed-signal CMOS chip which allows analog components to be connected together in an arbitrary fashion, allowing for rapid testing and measurement of many different circuit designs. The name RASP stands for Reconfigurable Analog Signal Processor.

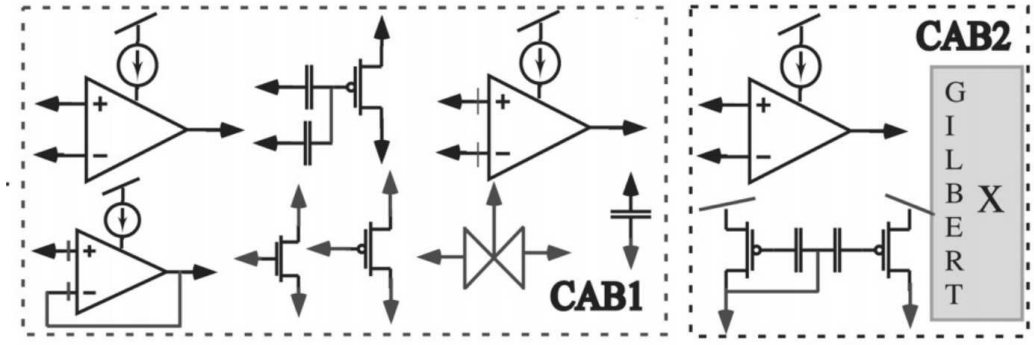
The FPAA is organized into three functional blocks. The first is the Computational Analog Block (CAB), which is a physical grouping of analog circuits which act as computational elements. These elements include nFETs, pFETs, Operational Transconductance Amplifiers, capacitors, Gilbert multipliers, and others. These CAB components can be connected together to form more complicated subcircuits, which can be further interconnected to create an analog computational system.

The interconnection of CAB components is accomplished with the FPAA's second functional block, the switch matrix. This is a collection of floating-gate pFETs which connect together rows and columns of routing lines. The switch matrices are arranged such that they allow local routing between elements inside a single CAB as well as global routing between CABs.

The third functional block is the programmer, which selects a floating-gate device in the switch matrix and controls the processes of tunneling and injection to add or remove charge to the floating gate. This allows each device to be turned completely on, turned completely off, or operated somewhere in-between. This flexibility means that switch elements can be



(a)



(b)

Figure 3: (a) The RASP 2.8a and 2.9a architecture. Computational Analog blocks are interconnected using floating-gate routing elements. Each floating-gate is programmed using a programmer and selection scheme. Analog input/outputs come in from all sides of the chip. Image modified from [3]. (b) Example of the components in Computational Analog Block. For this particular FPAA, there were two flavors of CAB, CAB1 and CAB2. Image from [3].

used for computation as well as routing, a benefit seen in other efficient routing applications [59], [36]. One example of a useful computational element created from floating-gates is a constant current source. A schematic representation of the FPAA is shown in Fig. 3(a).

1.5 Outline of the Dissertation

This dissertation contains contributions to reconfigurable analog signal processing and neuromorphic engineering. It consists primarily of two types of contributions – the implementation of circuits and systems on existing FPAA hardware, and the design and testing of new FPAA hardware. It is important for anyone working in this field to perform both tasks. To understand the capabilities of FPAAs, it is essential to grapple with current hardware and take notice of areas that could be improved. To move the state of the art forward, one must then take those lessons and apply them to the creation of new hardware. We translated testing to design most obviously with the neuromorphic chips from the CADSP lab. We will present implementation chapters first, for they provide insight into an FPAA's capabilities. Next we will present new FPAA designs.

Chapter 2 present's this dissertation's first novel circuit implemented on an FPAA – a transistor-only adaptive Winner-Take-All (WTA) circuit. A traditional WTA is essentially a "max" circuit. It has N inputs and outputs, and the only output current that is active is the one which corresponds to the input with the highest value. The adaptive WTA was a modification of the traditional WTA with extra dynamics to highlight changes in the input. This is much like attention in a neural system. The dynamics were implemented using a floating-gate transistor. In our implementation, we replace the floating-gate transistor with three pFETs – the active transistor and two other transistor which model tunneling and injection.

Chapter 3 discusses how a neurobiological structure known as a dendrite can be implemented on an FPAA. Dendrites are elements of a neuron which connect the neuron's input (synapses) to the neuron's cell body (soma). Dendrites which do not contain active ionic channels, known as passive dendrites, are typically modeled using a linear cable model. Chapter 3 shows how the circuit model of dendrites, the diffuser circuit, behaves

similarly to the linear cable model on the FPAA. The chapter also discusses nonidealities and implementation issues discovered when mapping the circuit to the FPAA.

Chapter 4 is also an FPAA application chapter. We discuss how a subtractive audio synthesizer can be implemented on the FPAA. We show that all the main components of a synthesizer can be easily implemented separately, and then we connect them together to create a full synthesizer. Besides implementing a new system on the FPAA, this chapter uses two important aspects of FPAA design – block-level system design and automatic circuit characterization. Previous work [54] centered around the creation of a Simulink tool that allows circuit designers to treat their subcircuits as blocks, enabling fast and easy system creation. We used that framework to create a complex synthesis system, demonstrating the promise discussed in [54]. We also wrote software to automatically characterize some of the circuits in the synthesizer, and we used those characteristics to improve performance.

Chapter 5 is the first application chapter focusing on neuromorphic hardware. Our lab's first large-scale neuromorphic chip, the Neuron1D, contains 100 neurons and 30,000 synapses. Similarly to Chapter 3, we wrote software to automatically characterize subcircuits in the Neuron1D, which was then used in actual testing. This characterization was critical to getting anything to work on the chip. If the synaptic current generation circuits had remained uncharacterized, the synapses would have been mismatched by several orders of magnitude in current, since the synapses are transistors operated in a subthreshold regime, where their current is an exponential function of their gate voltage.

Chapter 5 also contains an application focusing on learning in neuromorphic hardware. Neuromorphic chips contain synapses that learn from experience, and this should allow them to adapt their behavior to changing environments. In Chapter 5, we use a learning rule known as Spike-Timing Dependent Plasticity to perform simple learning experiments on the Neuron1D. The insights gained from this testing were then used to design the next neuromorphic chip, the Neuron1H.

Chapter 6 is the first design chapter. We designed, laid out, and tested a specialized DAC CAB in a new FPAA, the RASP 2.9v. The DAC CAB was then used for current-mode computation of image transforms.

Chapter 7 contains the bulk of the design contributions we made to reconfigurable circuits and neuromorphic engineering. Our lab created a new line of chips, the RASP 3.0 generation, which is unique for its inclusion of a synthesized microprocessor on each chip that acts as the main controller. We ported our floating-gate programmer to use the new microprocessor. Previously, programming of our chip was done with custom digital logic on-chip and an Atmel microprocessor off-chip.

Chapter 7 also discusses the 3.0-generation neuromorphic chip that we designed, the RASP 3.0H. We added a number of improvements to the 3.0H based on our testing of the Neuron1D. During the Neuron1D learning experiments, we found that STDP learning could be unstable, causing a single synapse to dominate the behavior of the entire chip. We added two features to counteract this instability. The simple fix was to add a cascode transistor in the path of the STDP injection transistor, so that the amount of injection is limited. The more complex fix was to add a homeostatic mechanism called synaptic scaling to the chip. Synaptic scaling effectively multiplies the sum of the inputs to a neuron by a constant in order to control the average spiking rate of the neuron. It is a feedback control mechanism that prevents neurons from spiking too fast or slow, and it has been noted to help control STDP's instabilities. We also added ionic channels that were not present in the previous chip, inhibitory synapses that follow an STDP rule, and on-chip biases.

Finally, Chapter 7 discusses the design of components for the RASP 3.0 RF FPAA. This chip was designed in 45nm, and its purpose is to perform signal processing at RF frequencies. Specifically, we hope to do beamforming and FIR filtering at RF frequencies. We designed an OTA-based delay element which will be used as part of the beamforming process. We also also briefly discuss the PCB designed for testing the RF chip, and we present initial test results.

CHAPTER II

A TRANSISTOR-ONLY ADAPTIVE WINNER-TAKE-ALL

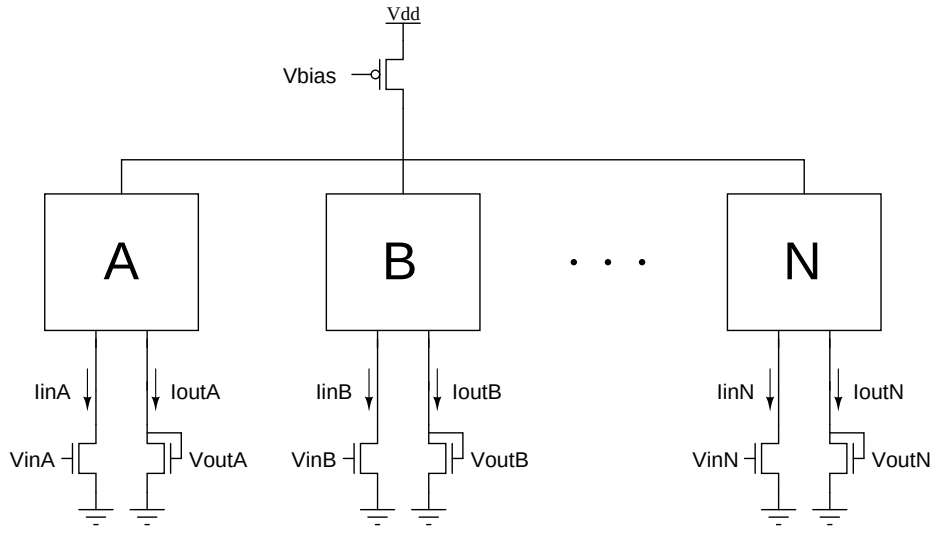
2.1 Introduction

A winner-take-all (WTA) is an array of cells, each with a single input and output [34]. Whichever cell has the largest input value will exhibit a large output, while all other cells will output almost nothing. The original WTA was implemented by Lazarro et al., who noted that this behavior mimics biology in that it has "general nonlinear inhibition" like groups of cells in the nervous system [35].

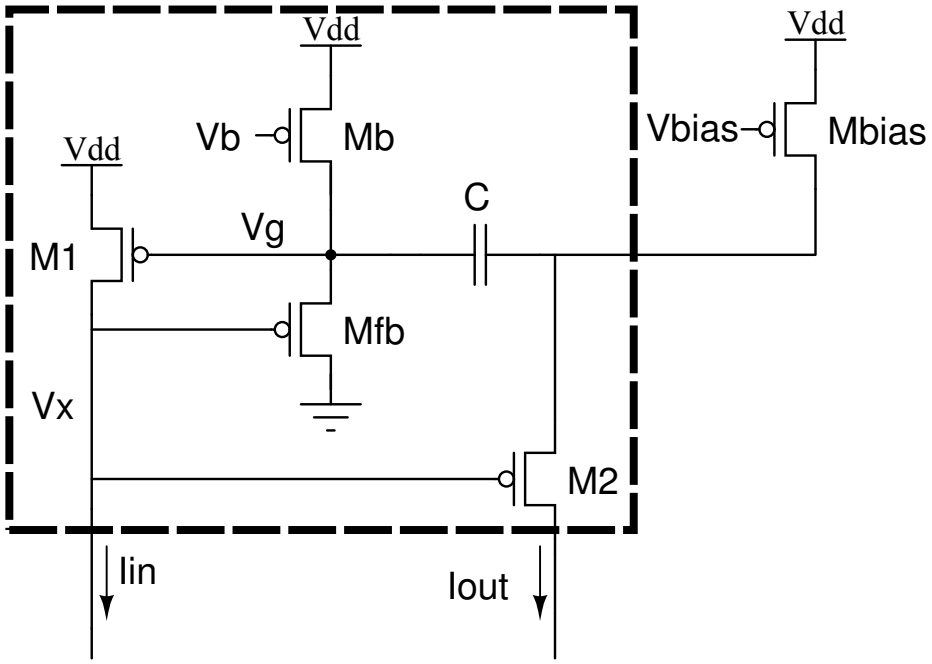
Kruger et al. modified this design using a floating-gate transistor to add transient dynamics to the system. The circuit would respond instantaneously to changes in the inputs, but its steady-state outputs would mimic a traditional WTA. This behavior is analagous to attention: if a car alarm goes off suddenly, the brain's attention is instantaneously turned to the alarm. If the alarm continues for several minutes, the brain will begin to ignore the sound.

Fig. 2.1 shows our circuit, which displays the adaptive behavior of [34] without the use of a floating-gate transistor. This approach has three advantages over the previous approach: it has improved adaptation times, can be implemented on a single-poly silicon process, and does not require the high voltage sources which are necessary for floating gates. We modeled the floating-gate transistor using three pFETs, a technique developed by Hasler et al. to model the autozeroing floating-gate amplifier using just transistors [24]. M_1 is the input transistor. M_b models electron tunneling, which makes the floating node positive in floating gate implementations. M_{fb} models hot-electron injection, which makes the node negative.

The circuit was implemented on a Field-Programmable Analog Array (FPAA) designed by ICElab. The RASP 2.8a was used [4]. This platform allowed for experimentation with different circuit configurations.



(a)



(b)

Figure 4: The experimental setup for our circuit. The input current is set using an nFET current source, and the output current is measured using a diode-connected pFET. The cells of the winner-take-all structure are represented by the lettered boxes. (b) The functional unit of the adaptive WTA structure (inside lined box). Mb and Mfb form a feedback network that adapts the output currents to a steady state. M1 and M2 form components of the traditional WTA.

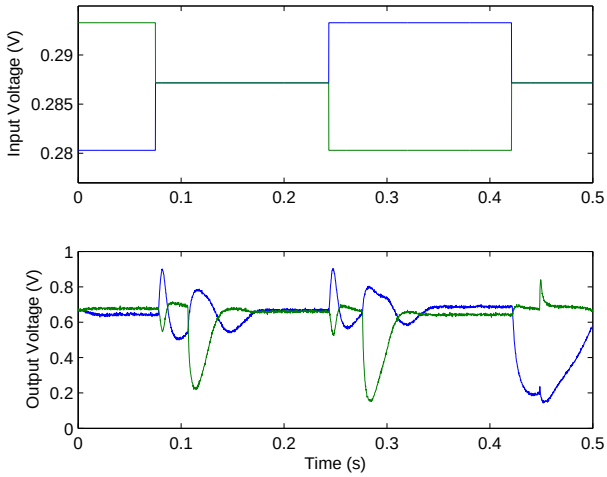


Figure 5: Step response of adaptive WTA for a two-input case. Both outputs exhibit a transient behavior upon a step input, but their steady-state behavior mimics a traditional WTA. In the steady-state, when the input currents are equal, the outputs are equal. When one current is larger than the other, its output current is much larger.

2.2 Dynamic Behavior

The dynamic behavior of the system is typified in Fig. 5. When the input to the system is instantaneously increased, the circuit displays a large second-order transient response. As time passes, the adaptive mechanisms begin to take effect, and a steady-state is achieved. We will show a method to create dynamics that are not underdamped in following sections.

The dynamics of this circuit can be analyzed by breaking up its response into two different regimes. The initial transient response has a time constant that is much faster than that of the adaptation, so the effects of M_b and M_{fb} can be ignored.

As time passes, the feedback transistors will begin to adapt the output to a steady-state value. It is useful to know the time constant for this adaptation. We derive the time constant by developin an equivalent circuit model. First, we notice that M_{fb} and M_b are connected in a source follower configuration. We model this as a buffer with a gain of κ and an output conductance g_s, which is set by V_b of Fig. 2.1. We also note that the node has a capacitance C_{eff} which is the sum of all capacitances seen at the node. The equivalent circuit is shown in Fig. 6. Assuming that the pole at the gate dominates the circuit's response, we write an approximation for the time constant:

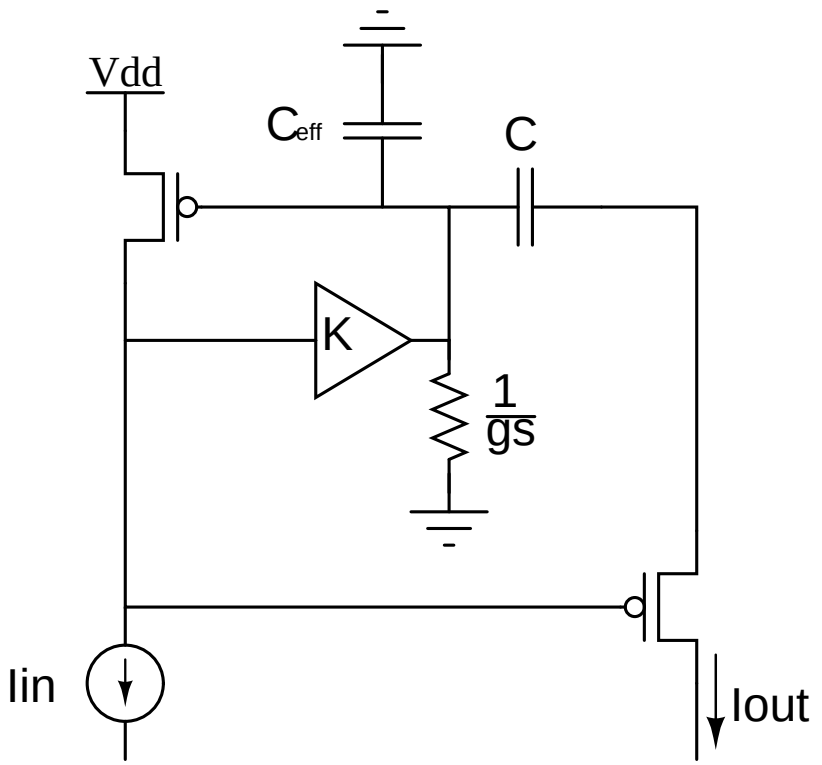


Figure 6: The equivalent circuit for operation when adaptation takes effect. The time constant of adaptation is set by the pole located at the gate, which is the effective capacitance of that node multiplied by the resistance. This pole can be moved by changing the bias current.

$$\tau = \frac{C_{eff}}{g_s} = \frac{C_{eff} U_T}{I_{bias}}. \tag{5}$$

This expression shows that τ is a function of the bias current set by V_b of Fig. 2.1. We show that this is indeed the case by varying the bias and measuring the circuit's step response. Fig. 7 shows that increasing the bias current reduces the length of time the circuit takes to adapt to a steady-state output. One other way to think about this behavior is that inreasing the currents through M_{b1} and M_{b2} allows the gate nodes of M_1 and M_2 to charge up more quickly, decreasing the transient response time.

2.3 Steady-State Behavior

In the steady-state solution, this circuit behaves like a regular WTA. We can make a few simplifications to the circuit diagram to ease the analysis. At low frequencies, the capacitor C acts like an open circuit. Transistors M_1 and M_{fb} form a current conveyor. This could

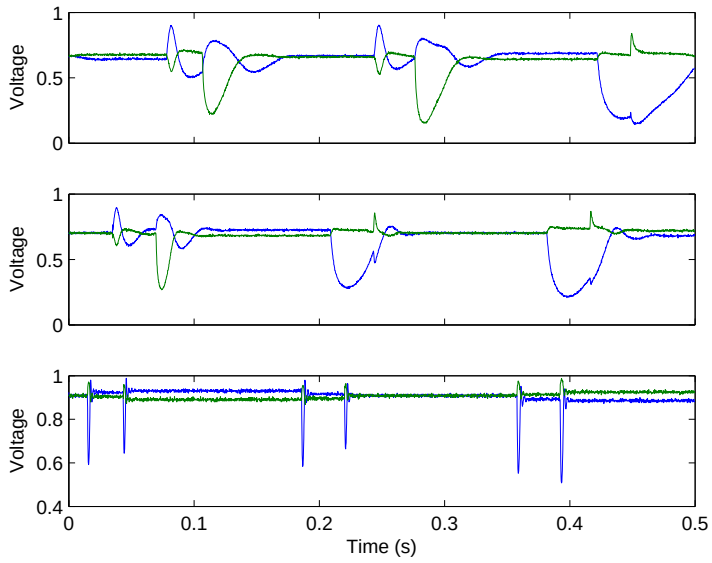


Figure 7: Increasing the bias currents decreases the amount of time it takes to charge the floating node. This will decrease the length of the transient response. The biases for the pFET current source of the top plot are 2.4 and 2.29V; for the middle plot, 2.3 and 2.15V; for the bottom plot, 2.1 and 2.04 V.

be modeled as a diode-connected FET with an increased body-effect (κ) value of κ^2. This simplified circuit is shown in Fig. 8.

The simplified circuit is a current normalizer. A current normalizer references many input currents to one bias current. The output currents of a normalizer have been shown to exhibit the following form [37]:

$$I_{out_i} = I_b \frac{I_{in_i}}{\sum_j I_{in_j}} \tag{6}$$

This expression provides simple intuition into the workings of the circuit. If a cell's input is large relative to all the other cells, its output will sink a greater portion of the output current.

We also note how the bias currents affect the output. We can derive the steady-state solution by noting that, at steady-state, the current through M_b must equal that through M_{fb} and the input current must equal the current through M_1. Applying Kirchhoff's Current Law (KCL) at V_g and V_x yields the following two equations:

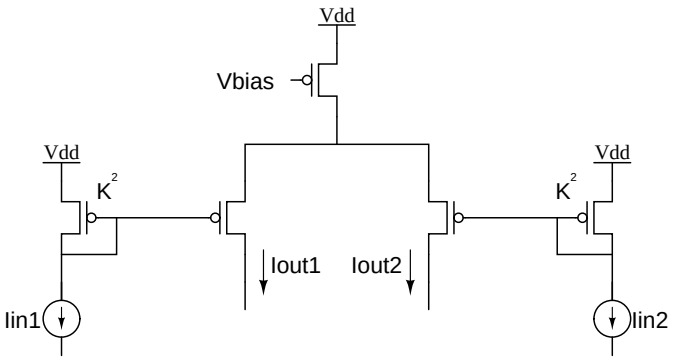


Figure 8: The simplified version of a two-input adaptive WTA. This behaves like a current normalizer, which yields output currents weighted according to the size of their input currents.

$$I_b = I_0 e^{\frac{V_g - \kappa V x}{U_T}} \tag{7}$$

$$I_{in} = I_0 e^{\frac{V_{DD} - \kappa V_g}{U_T}}. \tag{8}$$

V_g in Eq. 7 refers to the node we have named V_g and acts as the source of M_{fb}. In the case of our two-input WTA, these two equations lead to an expression for the difference between the V_x values V_{x1} and V_{x2} as

$$V_{x1} - V_{x2} = \frac{U_T}{\kappa^2} \ln \left(\frac{I_{in1} I_{b1}^{\kappa}}{I_{in2} I_{b2}^{\kappa}} \right). \tag{9}$$

Since the M_2 transistors from the two subcircuits form a differential pair, the difference between V_{x1} and V_{x2} determines which input "wins." Equation 9 shows that the DC biases and the input currents affect the winner. We have experimentally verified this in Fig. 9: as we vary the bias of cell 1 relative to cell 2, the DC levels of the outputs drift. In an extreme case, if the biases are set at significantly different values one input will always win regardless of whether its absolute value is greater or less than the other input.

This dependence on DC bias point could be a benefit. It allows users to control how important one type of input is relative to another. Reverting to the attention example, an ambulance's siren may be more important than a dog's bark. On the other hand, this dependence on bias means that users have to carefully set each bias point, which could take

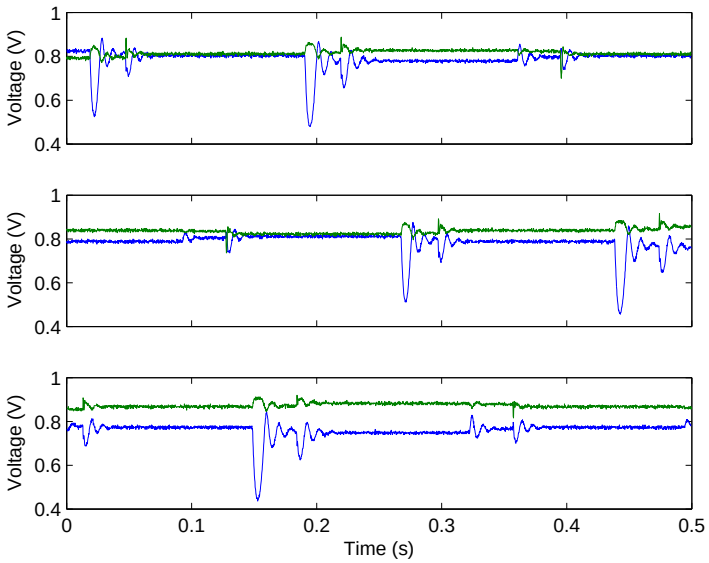


Figure 9: The bias currents affect the DC offset of the steady-state outputs. For all three plots, Vbias1 is kept constant at 2.2V. Vbias2 is decreased from 2.14V at the top to 2.12 V in the middle to 2.10V at the bottom. While both could win when the biases were set correctly at the top, one output always wins when the biases are not set correctly, as shown at the bottom.

a lot of time for a many-input case.

2.4 Variations on Initial Circuit

The FPAA platform allowed experimentation with the circuit's implementation. The first change we made was to add extra capacitance between the gate nodes and the common node V. Earlier we predicted that the time constant of the circuit is dependent on the total capacitance at the node, so we expect that more capacitance would increase the time constant. Our results are shown in Fig. 10.

We also implemented a four-input version of the adaptive WTA. Its response is shown in Fig. 11. Its steady-state behavior is as we would expect: when one input is greater than the others, its output current is also greater. One unexpected result is that all cells exhibit a transient behavior even if just a few are changed. Some of this may be a result of capacitive feedthrough from the changing inputs to nodes in the other cells.

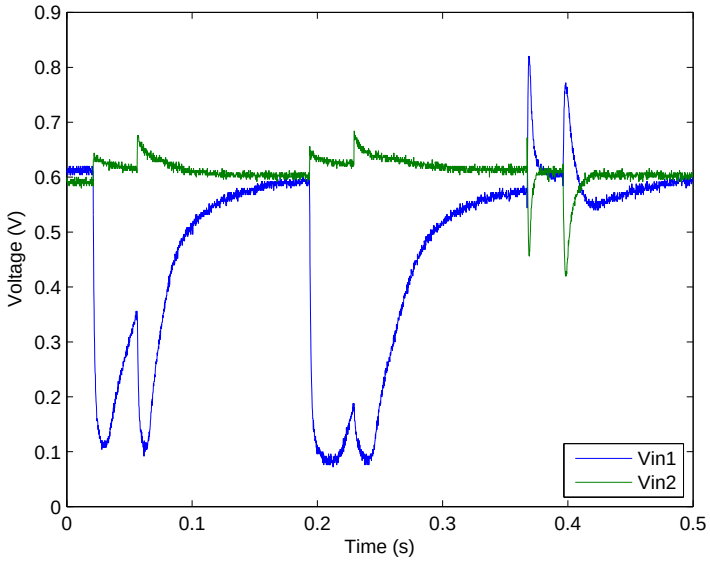


Figure 10: Step response of adaptive WTA with extra capacitance between the input transistor's gate node and the common node. The most obvious effects of this change are that the circuit does not display ringing behavior in its transient response, and its time constant is much slower than the first case. This makes intuitive sense because there is more capacitance to charge up on the gate node, so the same bias current will take longer to charge it.

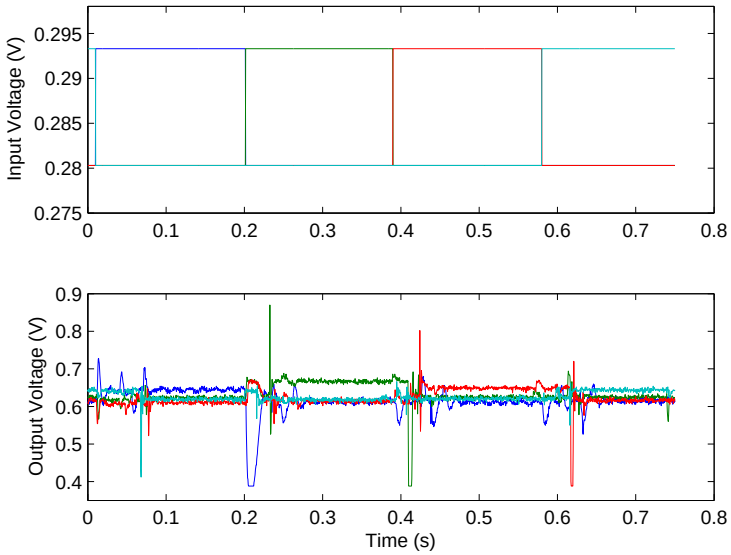


Figure 11: Response of an adaptive WTA with four inputs. Although it is difficult to see on this plot, upon each transition all four outputs show some transient response, even though just two inputs change at a time. This could be the effect of capacitive feedthrough.

2.5 Conclusion

We have implemented multiple realizations of an adaptive WTA using a reconfigurable analog platform. The dynamics of the circuit are a useful analogue to attention in nature: immediate input changes yield transient changes in the output, but in time a less-excited steady-state behavior prevails. The transient behavior can be changed by modifying the capacitance at important nodes in the circuit or changing the bias current.

CHAPTER III

MODELING DENDRITES ON A FIELD-PROGRAMMABLE ANALOG ARRAY

This chapter explores the equivalences between biological dendrites and CMOS dendrites, and it discusses the implications of performing these tests on a reconfigurable platform.

3.1 Introduction to Dendrites

An important structure in biology is the dendrite, a highly-branched conductive medium that connects the neuron's synapses to its soma, as shown in Fig. 12.

For many years, neuroscientists assumed that dendrites did not add much significant computational value to networks in the brain. They were modeled simply as wires by researchers in artificial neural networks. Recently however, a number of neuroscience reviews have posited that dendrites and single neurons have more computational power than previously believed [32], [40]. To take advantage of this computation, we have verified that some of the most basic properties of dendrites such as their steady-state decay and can be observed using analog CMOS circuit models.

There is a long history of dendritic emulation in the neuromorphic community. One of the first projects was undertaken by Elias [16], who created compartmental models consisting of resistors of various layouts, capacitors, and synapses implemented by MOSFETs. He demonstrated spatial weighting of inputs, sublinear summation of nearby synapses, and tonic summation of inputs. He implemented simple computational systems such as a signal symmetry detector and a direction selectivity system.

Another classic dendritic implementation was completed by Rasche and Douglas [49]. They chose to implement the axial conductances with switched capacitors and the leakage conductances with OTAs. They performed extensive tests on how cable properties change as a function of the conductances, boundary conditions, and compartment lengths. They also observed how action potentials propagated down the passive cable and demonstrated

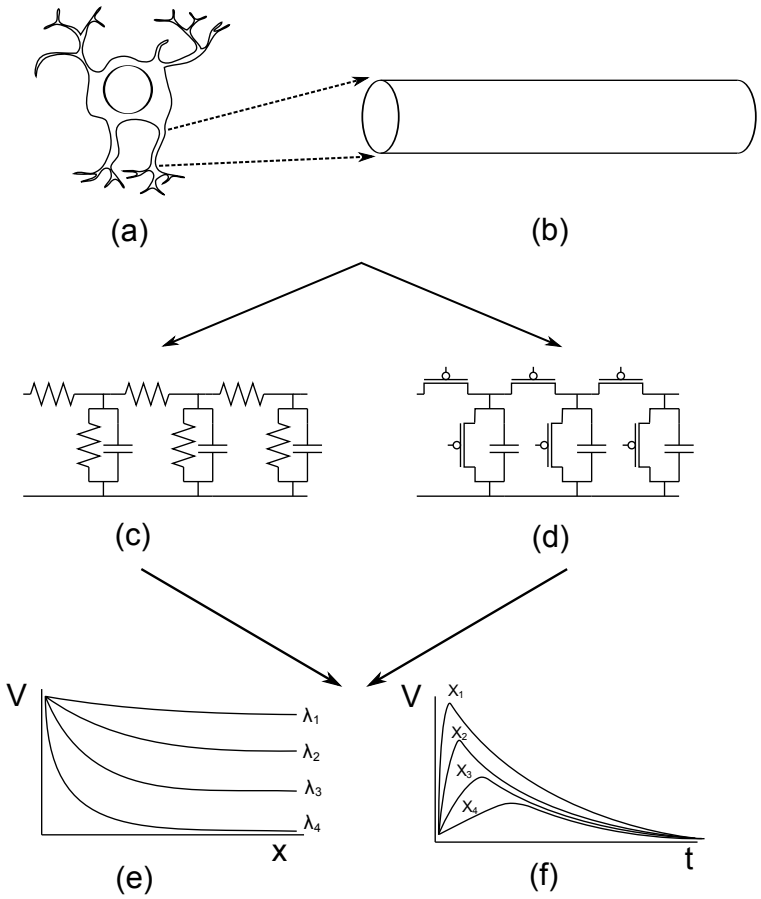


Figure 12: When operated in the correct regime, a VLSI dendrite model produces the behavior predicted by canonical linear models. (a) Dendrites are the structures that connect synapses to the cell body. They perform linear (and sometimes nonlinear) summations of input currents. (b) Neuroscientists typically model these structures as passive linear cables. (c) The classical model for this linear cable is an equivalent RC delay line. The major predictions of linear cable theory are based on this model. (d) An alternative model for the linear cable is a network of aVLSI elements, primarily MOSFETs and capacitors, where input currents are translated into small voltage signals which swing around a DC operating point. If (c) and (d) are equivalent, they should behave similarly. (e) The steady-state behavior of both models is expected to be an exponential decay in voltage, where the amount of decay depends on physical parameters. (f) The dynamic behavior of both models is expected to be exponential decay in space and a delay in time.

that the cable could act as a directionally sensitive system.

A more recent review was completed by Wang and Liu [60]. Their system consisted of computational subunits which included nonlinear synapses, a spiking circuit, and a cable connecting the compartments together. They investigated how the response of the dendrite changes based on the spatiotemporal pattern of the inputs to the system. They

emulated N-methyl-D-aspartic acid (NMDA) channels, which are ligand-gated channels for the neurotransmitter glutamate [31]. They showed how activating NMDA channels leads to superlinear responses in the system and that these nonlinearities allow the dendrite to discriminate between input patterns with different spatial extents.

This chapter's primary focus is to discuss the benefits and drawbacks of a simple dendrite model implemented in a reconfigurable environment. Our model consists of P-channel MOSFETs operating in their linear regime. We choose this topology because it lends itself well to scaling in a reconfigurable environment. We run simple experiments that demonstrate the model's equivalence to a classical linear cable, and then we discuss nonidealities caused by the reconfigurable environment.

This chapter is organized as follows. First, we develop a small-signal circuit model for the dendrite. Next, we present both static and dynamic measurements from these circuits and compare them to cable theory, as described by Rall's linear cable model. Finally, we provide a brief overview of extensions of the basic cable model, such as nonlinear behavior of the circuits and implications of placing them in large-scale analog systems.

3.2 Implementing the Linear Cable Model with Analog CMOS Circuits

Historically, dendrites have been modeled as linear cables. Their structure consists of a conductive solution that allows current to flow from the synapse to the cell body; a phospholipid bilayer which separates the membrane potential from the external potential; and ion channels which allow small amounts of current to leak across the membrane. Wilfrid Rall adapted the mathematics originally developed to model core conductor cables and applied it to dendrites [55]. We will demonstrate that the behavior of a CMOS dendrite with pFET channels reduces to Rall's mathematical model when operated with small-signal inputs.

Our thesis is shown in Fig. 13. We begin with the biological dendrite and model both the conductive medium and the leak channel using a silicon channel. We also provide a bias current to set the resting membrane potential, V_{rest}. We then assume small signals are applied as inputs, and our circuit reduces to a linear model.

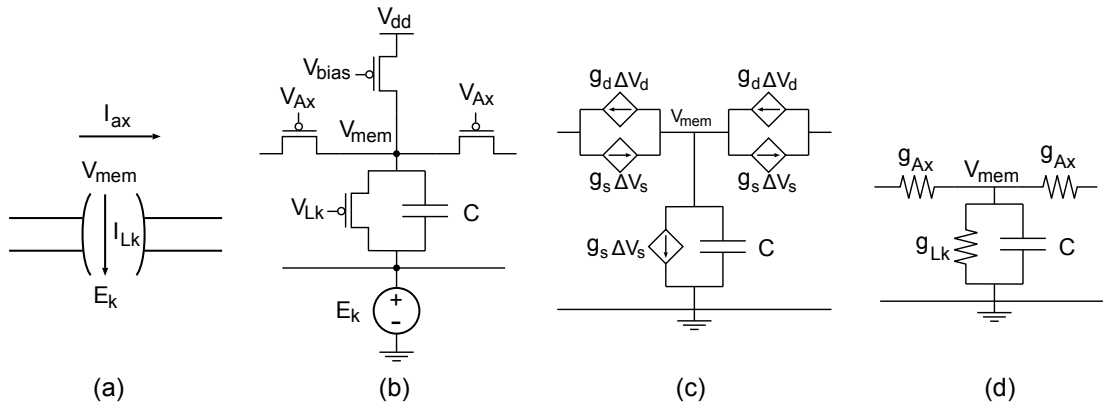


Figure 13: Various models of a dendrite. A biological dendrite is modeled as a conductive cylinder surrounded by an insulating layer. A cross section of this model is shown in (a), where I_{ax} represents the current flowing along the axial direction of the dendrite, I_{Lk} represents current from the dendrite to extracellular fluid through a leak channel, and the internal and external potentials are V_{mem} and E_k, respectively. When we translate channels into transistors, we get the model shown in (b), where both the axial and leakage current flow through transistors. The external voltage is set by a voltage source E_k, and V_{mem} is set by the bias structure. When we linearize the transistor model, the result is shown in (c) and (d). Current sources can be reduced simply to small-signal conductances.

3.2.0.1 Introduction to Linear Cable Theory

The simplest model neuroscientists use to describe the function of dendrites is known as the Linear Cable Model. The dendrite is treated as a conductive core surrounded by an insulating layer. The core is modeled as a long piece of resistive material, which can be discretized into many incremental resistances R_{Ax}. The insulating layer is a phospholipid bilayer, and it is modeled as a capacitance C because it separates the internal membrane potential from the extracellular potential. However, there is leakage current from the intracellular solution to the outside of the cell, so a leakage resistance R_{Lk} is also included in the model.

Koch gives a simple derivation of the mathematical cable model for this circuit in [31]. If one writes down Kirchhoff's Current Law (KCL) at the nodes V_{mem} and uses Ohm's Law $V = IR$ and the capacitor equation $I = C\frac{dV}{dt}$, then the following differential equation describes the system:

$$\lambda^2 \frac{\partial^2 V_{mem}}{\partial x^2} = \tau \frac{\partial V_{mem}}{\partial t} + V_{mem} - R_m I_{inj}, \tag{10}$$

where I_{inj} is current injected into the dendrite, $\tau = R_{Lk}C$ and $\lambda = \sqrt{\frac{R_{Lk}}{R_{Ax}}}$. τ and λ are called the time constant and the space constant. Intuitively, τ determines how voltages along the dendrite change with time, and λ determines how voltages change with distance down the dendrite. If we only care about the steady-state solution, we can set the differential with respect to time equal to zero. This results in a solution for the steady-state behavior given in Eq. 11.

$$V(x) = V_0 e^{-|x|/\lambda} \tag{11}$$

3.2.0.2 Using Silicon Channels to Implement the Linear Cable Model

Our goal is to replace the resistances in the linear cable model with silicon channels. The most intuitive way to do this is to simply replace each resistance with a single pFET. The axial resistances are replaced with a pFET whose gate is set at a fixed potential, V_{Ax}. Similarly, the membrane resistances are replaced with pFETs whose gates are set at a fixed potential V_{Lk}. On an intuitive level, the conductance of the pFETs is set by their gate voltage. We will need to bias the dendrite at a fixed membrane potential, so a transistor which provides a DC bias current is inserted into each node of the dendrite. This could be seen as analagous to ion pumps in the membranes of biological channels. The bias transistor has a gate voltage V_{bias}, and it sets the DC point V_{mem}. The final piece of the dendrite to consider is the capacitance. Every node of analog circuits has some capacitance associated with it. So we do not have to place an explicit capacitance at each node to simulate a dendrite. If we so desire, the FPAA has the ability to compile 500 fF capacitances into the nodes. The final circuit is as shown in Fig. 13b.

To model an equivalence to the linear cable model, we can simplify the full circuit into a linear one. Each transistor is replaced with a small-signal, linearized model. To do this, we take partial derivatives of the current equation for a pFET as formulated in Eq. 2.

3.2.0.3 Linear Model of Axial FET

In the operation of the circuit, we will leave the gate fixed at a DC bias, so we can simplify Eq. 2 by incorporating the gate voltage term into $I_{bias} = I_0' e^{-\kappa V_g/U_T}$. Therefore, the current

through the axial and leakage pFETs can be expressed as follows:

$$I = I_{bias}\left(e^{V_s/U_T} - e^{V_d/U_T}\right) \tag{12}$$

Traditionally, we form a linear model for this device by taking the partial derivative of the current with respect to a changing terminal voltage. Since a signal is traveling in the axial direction of our dendrite, both the source and the drain of the axial FET are changing. We model this with two current sources in parallel pointing in opposite directions, with the values $g_s\Delta V_s$ and $g_d\Delta V_d$. Ignoring channel length modulation, the values for g_s and g_d are given in [37] as:

$$g_s = \frac{\partial I_{Ax}}{\partial V_s} = \frac{I_{bias}}{U_T}e^{V_s/U_T} \tag{13}$$

$$g_d = -\frac{\partial I_{Ax}}{\partial V_d} = \frac{I_{bias}}{U_T}e^{V_d/U_T} \tag{14}$$

Note that, at rest, the dendrite will be biased such that all source and drain nodes of the axial pFETs will be at the same rest potential, V_{rest}. This means that $g_s = g_d$. We can combine the two current sources into one source with the value

$$I = g_{Ax}\Delta V_s - g_{Ax}\Delta V_d \tag{15}$$

$$= g_{Ax}\left(\Delta V_s - \Delta V_d\right) \tag{16}$$

$$= g_{Ax}\Delta V_{sd} \tag{17}$$

So this is simply a small-signal conductance,

$$g_{Ax} = \frac{I_{bias_{Ax}}}{U_T}e^{V_{mem}/U_T} \tag{18}$$

3.2.0.4 Linear Model of Leakage FET

Modeling the leakage transistor is much easier. Both the gate and the drain are fixed to DC voltages. So any change in voltage across the device is completely due to a change in

the source. Therefore, the small-signal conductance of the leakage FET is just the source conductance, as given above:

$$g_{Lk} = \frac{I_{bias_{Lk}}}{U_T} e^{V_{mem}/U_T} \tag{19}$$

3.2.0.5 Deriving the Space and Time Constants

The space constant is the parameter λ in the linear cable equation which describes how voltage in the dendrite decays with position along the dendrite. It is related to the ratio of the axial and leakage conductances. Now that we have linearized our model, we can define a space constant λ by taking the ratio of our conductances:

$$\begin{aligned} \lambda \equiv \left(\frac{R_{Lk}}{R_{Ax}}\right)^{1/2} = \left(\frac{g_{Ax}}{g_{Lk}}\right)^{1/2} = \left(\frac{I_{bias_{Ax}}}{I_{biasLk}}\right)^{1/2} \\ = e^{\frac{\kappa(V_{Lk}-V_{Ax})}{2U_T}} \end{aligned} \tag{20}$$

Fig. 14 verifies this expression experimentally using the FPAA. We measured how the conductance of a pFET changes as a function of its DC gate potential. To relate this back to Eq. 20, we measure a reference conductance and see how changing the gate voltage affects the square root of the ratio of the new conductance to the reference.

The time constant τ describes how voltages decay with time. It is defined as the product of the leakage resistance and the capacitance, or

$$\tau = \frac{C}{g_{Lk}} = \frac{CU_T}{I_{bias_{Lk}}} e^{-V_{mem}/U_T}. \tag{21}$$

3.2.0.6 Sources of Error

The above expressions hinge on perfect matching among all pFET devices. This unfortunately is rarely achieved. We measured the values of κ and I_0 for a sample of 15 pFET CABs in the FPAA and measured the statistical variation for these two parameters. This information is shown below:

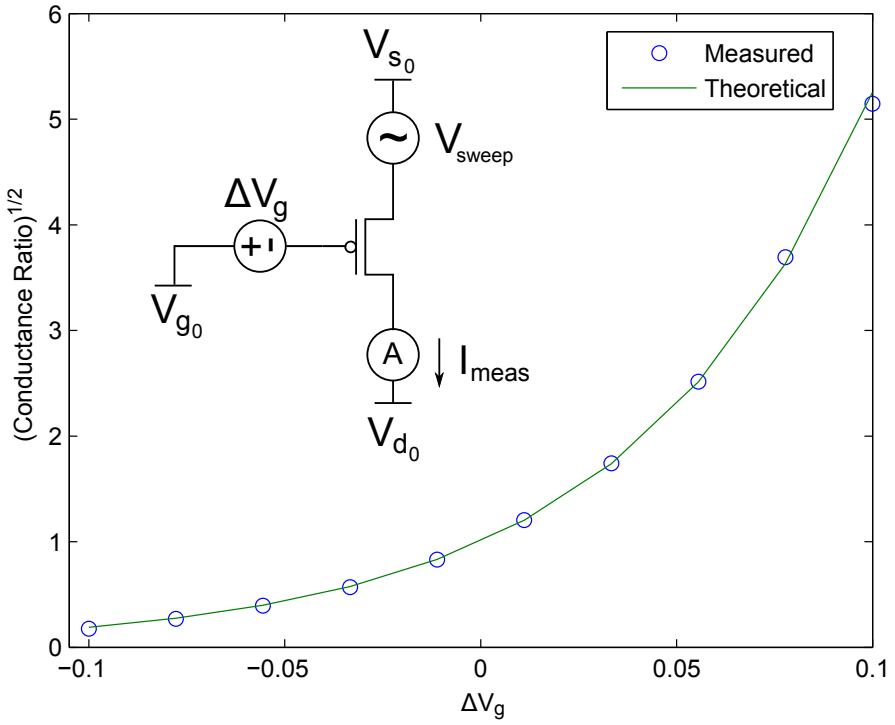


Figure 14: Demonstration that the ratio of source conductances is a function of the difference between gate voltages. We took a CAB pFET and measured a reference source conductance by fixing the DC potential at all of its terminals (V_{s_0}, V_{g_0}, and V_{g_0}), and measuring the DC current. We then swept its source voltage through a small range (V_{sweep}) and measured the change in current. The reference conductance was the slope of change in current with respect to change in source voltage. We performed this same experiment for ten different values of the gate voltage (V_{g_0} - ΔV_g). We then plotted the square root of the ratio of source conductances as a function of the gate voltage. We used the difference in gate voltages to create a theoretical value of the conductance ratio from Eq. 20, and the two match closely.

	μ	σ
κ	0.8393	0.0021
I_0	4.5740 fA	0.77549 fA

The above analysis assumes the system is processing "small" signals. We can no longer assume that the linear models behave if they are perturbed far from the DC bias. We limited inputs to the system such that the source nodes of the vertical pFETs never changed by more than 25 mV.

3.3 Demonstrating Equivalence to the Linear Cable Model

We now demonstrate that our voltage-mode circuit retains many of the behaviors of a passive dendrite. We set up our cable using the system shown in Fig. 15(a).

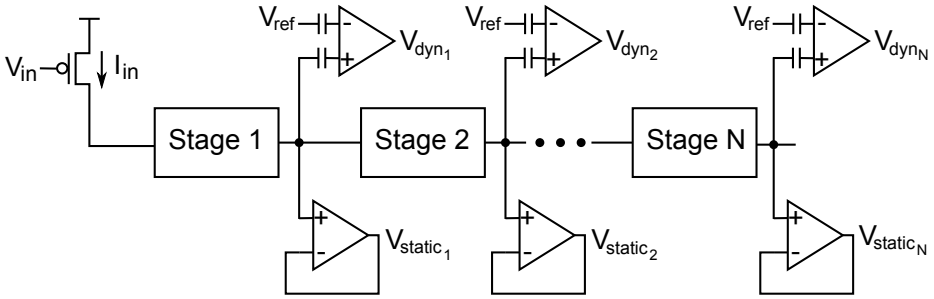
3.3.1 Steady-State Experiments

We first perform a steady-state analysis. We compiled a 10-stage dendrite onto the FPAA. We set $E_k = 1V$ and biased the membrane voltage to around 20 mV above E_k. Due to mismatch among the bias transistors and leak transistors, not all membrane voltages were exactly the same, and they could vary by as much as tens of mV. We attempted to compensate for some of the mismatch by an iterative process of measuring and changing the bias voltages on the gates of the I_{bias} transistors, but this did not remove all of the mismatch. Since this is a dendrite of finite length, the steady-state solution takes on a slightly different form than that given earlier. From [31], the solution is

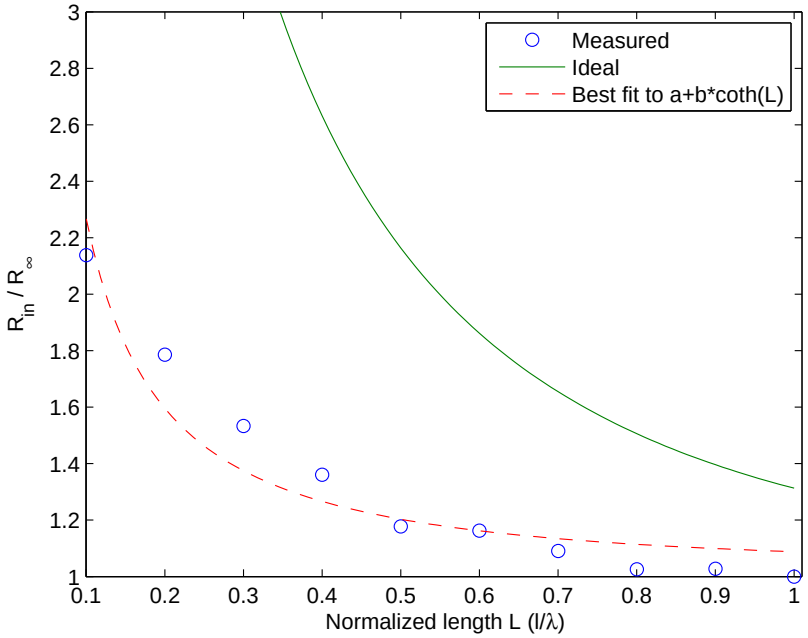
$$V(X) = V_0 \frac{\cosh(L - X)}{\cosh(L)}, \tag{22}$$

where X = x/λ and L = l/λ. For this experiment, we defined the steady-state voltage of a particular node as the difference between its measured rest voltage and its voltage after applying an input. The results for this dendrite are given in Fig. 16.

The input resistance of a semi-infinite, sealed-end cable is also well-known. Its expression is given in [31] as

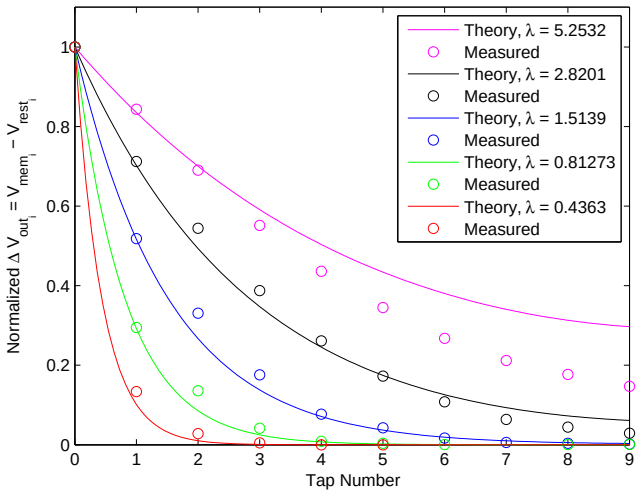


(a)

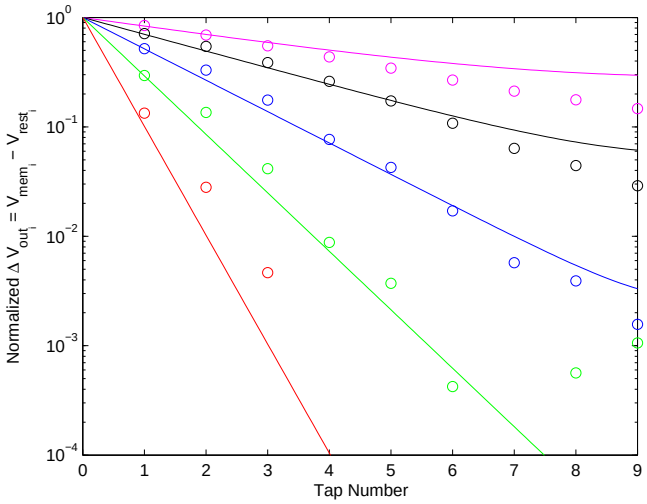


(b)

Figure 15: (a) Schematic for taking measurements. Each block representing a stage consists of one bias, axial, and leakage transistor (13b). At the output of each stage, two amplifiers relay the signal to a mux. The first is an open-loop floating-gate OTA, used to measure step responses at each stage of the dendrite. The second is a buffer-connected OTA, used to accurately read DC voltages for steady-state experiments. (b) Input resistance changes as dendrite length is increased. A fixed input current was injected into node 1, and the membrane voltage was measured before and after injection. We then calculated the difference between these two ($V_{delta} = V_{mem} - V_{rest}$). This was done for many different dendrite lengths. To calculate R_{in}/R_{∞}, we divided all values of V_{delta} by the value for L = 1. Since the injected current was the same for all tests, the ratio of resistances is therefore the ratio of the voltage responses. The response did not follow the quantitatively predicted curve, but it does demonstrate qualitative behavior similar to what we expect, as shown by the dashed curve fit to a + b*coth(L).



(a)



(b)

Figure 16: (a) and (b) Steady-state decay of dendrite voltage. For five different values of λ, a ten-stage dendrite was biased at DC such that the V_{mem} nodes were about 20-50 mV above $E_k = 1$. Then a small DC current was injected into the first node. We measured $\Delta V_i = V_{mem_i} - V_{rest_i}$ for every node in the dendrite. Then ΔV_i was normalized. Dots are experimental measurements, and lines represent how the voltages should decay if λ matches the theoretical value perfectly. The theoretical values essentially predict the "slope" of the logarithmic response, and not the actual DC offset. This is why the normalized predictions are accurate for low values of stage number and seem to deviate as stage number increases. We're seeing error in the slope but not DC offset. The linear plot gives an intuitive physical feel for how the dendrite behaves, while the logarithmic plot demonstrates how these are approximately exponential responses and how error in slope accumulates. Note that any changes which were negative (all of which were small) are not shown on the log plot.

$$R_{in} = R_{\infty} \coth(L). \tag{23}$$

As L increases, R_{in} approaches R_{∞}. To test whether our dendrite follows this model, we applied a step input current of I_0 to our dendrite and varied the value of λ. For a fixed input current but variable dendrite length, we can predict what the voltage should be at various points along the dendrite. Our results are shown in Fig. 15(b).

Our theoretical results do not perfectly match the data, and there are a few possible reasons for this. Probably the largest contributor to the problem is biasing the dendrite correctly. For the experiments in Fig. 16 a,b, the resting membrane potentials were as much as 30 mV away from each other. The ratio of small-signal conductances is $e^{(\Delta V/U_T)}$, so this means that the ratio of two ideally matched conductances could be as high as 3.32. It should also be noted that κ changes with the source voltage, so a 30 mV mismatch in source voltage could also affect κ.

3.3.2 Dynamic Experiments

Cable theory provides us with a prediction for what the shape of the step response should look like at the site of current injection. The form is given in [31] as

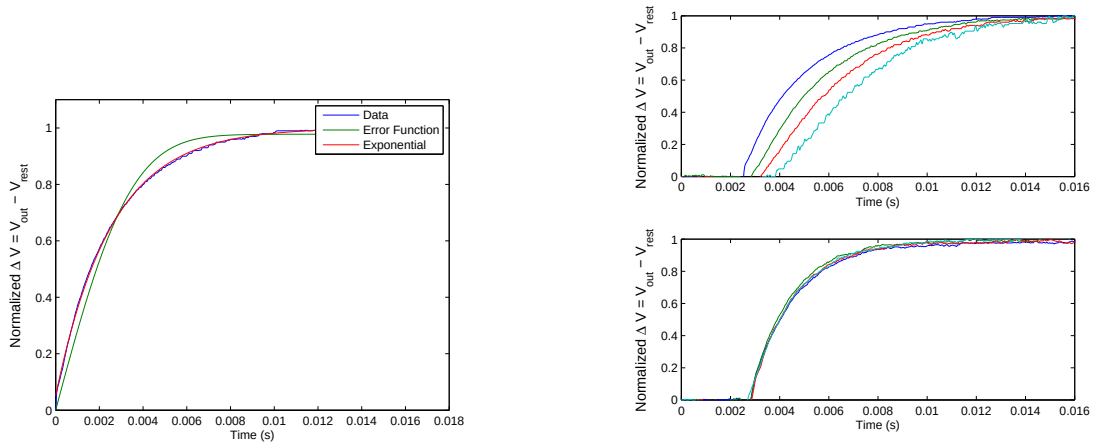
$$V_{Step}(0,T) = \frac{I_0 R_{\infty}}{2} erf(\sqrt{T}). \tag{24}$$

We have plotted a representative step response for x=0 along with a best-fit line to this theoretical function in Fig. 17a.

Since the cable model is basically an RC network, we expect to see delay down the line. The propagation velocity of a step input down the line is given in [31] as

$$v = 2\frac{\lambda}{\tau}. \tag{25}$$

This means that we can increase the delay down the line (decrease the velocity of propagation) by decreasing λ or increasing τ. In our experiment, we changed λ and looked at how the velocity of propagation was affected. The results are shown in Fig. 17b.



(a) (b)

Figure 17: (a) Step response for the first node of a diffusor, along with the best fit to the error function and an exponential function. The step response was obtained by setting the value of V_{ref} on the first node's floating-gate OTA such that V_{dyn_1} was midrail. Then the input current was pulsed, and the waveform was captured. We experimentally determined how much to pulse V_{in} by alternatively pulsing it, measuring how much the first node's voltage changed, and adjusting the gate until the first node's voltage changed by less than U_T, or 25 mV. We chose this value since the FETs would leave saturation if the source voltage changed by much more. We normalized the result by subtracting the DC offset and dividing by the maximum value reached. Linear cable theory predicts that the error function will be a closer fit than the exponential, but the data for our system mirrors an exponential response much more closely. It is possible that our step size was greater than needed to keep all devices in their linear regimes. (b) Step responses for four taps of the dendrite were taken for two different values of λ. For a small value of λ, the velocity of propagation is small, so one can see delays of the response as they travel down the dendrite. For higher values of λ, the velocity of propagation is fast, so little delay can be seen. Fig. 18 shows parasitic transients not visible in this figure.

In both the steady-state and dynamic experiments, we have seen a trend in our results. Namely, they agree with cable theory qualitatively but do not match it quantitatively. We do not expect these nonidealities to affect usability of the dendrites greatly. This is because we predict the computation in dendrites is not governed by precise tuning of every parameter. Neural computation is inherently different from the von-Neumann architectures in which precision is key. They exhibit high levels of stochastic behavior, redundancy, and recurrent connections. Rather, it was more more important to see that the basic dendritic properties can be varied over a wide range, allowing gross tuning of parameters.

3.3.3 Effects of a Reconfigurable Testbed

A reality of working in a reconfigurable environment is that parasitics can cause nonidealities. Fig. 18 demonstrates this. To apply an input current to our system, the gate of a pFET is pulsed low. This pulse can capacitively couple both into the system and into the instrumentation measuring the system's response. The amount of coupling depends on how the system is routed, so care should be made to ensure that system components are routed to minimize such effects. For instance, the routing lines for the voltage measurement circuitry should not be physically close to the digital pulse on the gate of the input current source. Additionally, a cascode should be used on the input current source.

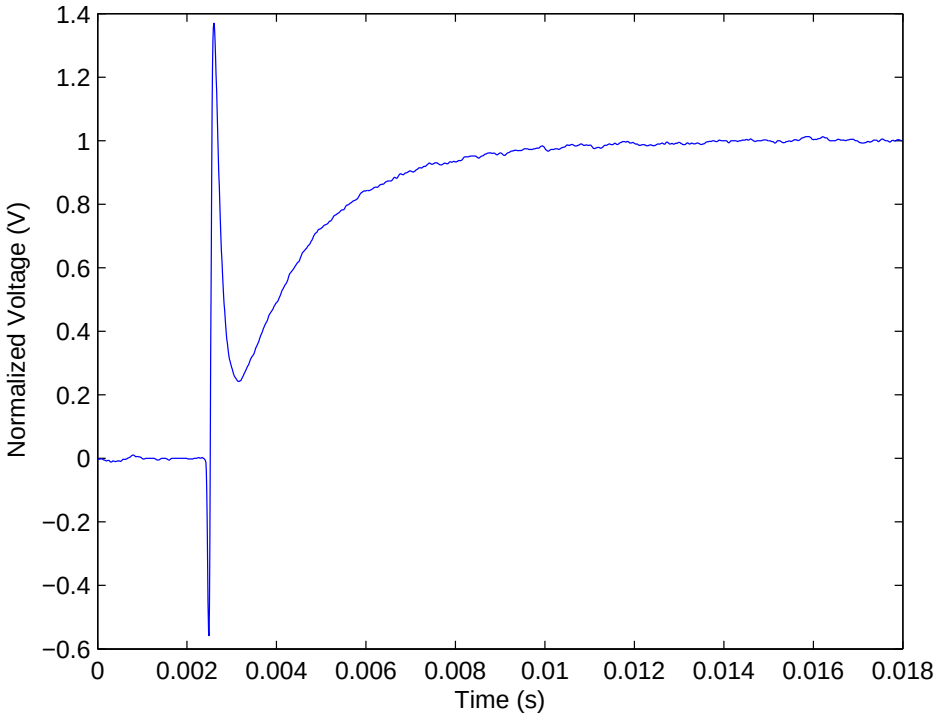


Figure 18: Two parasitic effects seen at once for one particular step response. When the gate of the pFET is pulsed down, some of that voltage change is coupled into the input node of the dendrite, and therefore initially the voltage at the membrane decreases. This change can be seen propagating along the system. For this step response, we also see a spike upwards. This is likely due to capacitive coupling into the instrumentation amplifier (a floating-gate input OTA), because this change is not seen propagating down the dendrite in other plots.

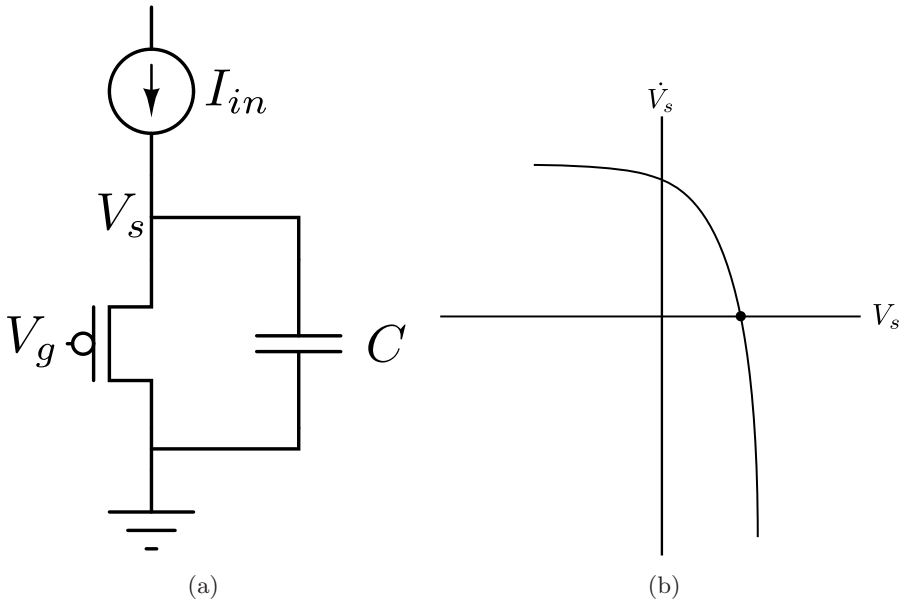


Figure 19: (a) Illustration of nonlinear dynamics in dendrite circuit. A large-signal input current is sent into a node which sees a transistor and capacitor in parallel. (b) Illustration of the phase portrait resulting from the circuit in Fig. 19(a). The input current moves the line vertically, which changes the qualitative behavior of the system.

3.4 Nonlinear Behavior of Dendrites

Most of this chapter has concerned the behavior of the dendritic circuit operated in its linear regime. When the input current becomes large, however, the qualitative behavior of the circuit changes, and nonlinear effects begin to take hold. Typically, a difference between drain and source of about $4U_T$, or 100 mV, is considered the nonlinear regime of the dendrite. To get a qualitative understanding of the nonlinear effects, we will analyze one "section" of dendrite, shown in Fig. 19(a).

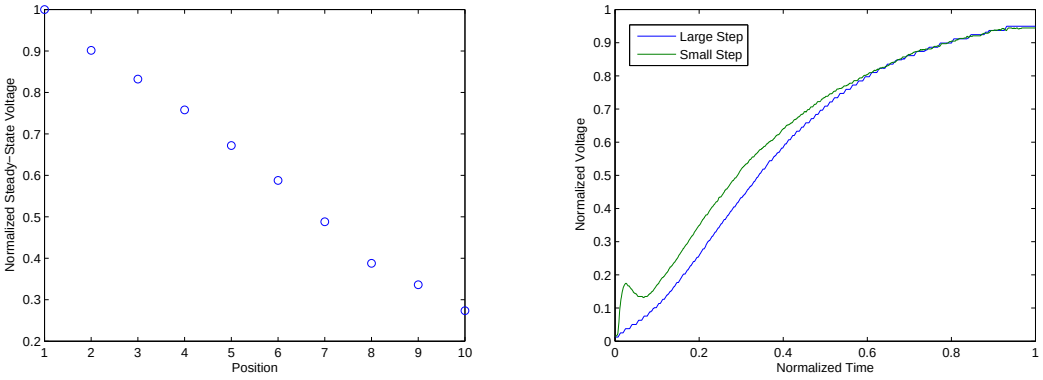
3.4.1 Math Modeling

Applying KCL and the current equations for a capacitor and a saturated transistor, we get

$$\frac{dV_s}{dt} = \frac{I_{in}}{C} - \frac{I_{bias}}{C} e^{V_s/U_T}. \tag{26}$$

We can use Eq. 26 to plot a phase portrait. The basic shape is a negative exponential with a vertical offset, shown in Fig. 19(b).

This portrait gives us quantitative and qualitative information about our circuit's voltage



(a) (b)

Figure 20: (a) When the steady-state response of a 10-stage dendrite is measured with a large input current (causing a change of about 200 mV at the first node), the response is a linear degradation in voltage. (b) Comparing shapes of small step and large step response. The step response was normalized in voltage by dividing by the steady-state value, and time was normalized by finding the point at which the voltage rises to 95% of its steady-state value. The initial response of the small step is more of an RC response, while the large step shows a sigmoidal behavior. See Fig. 18 for a discussion of the transient at the beginning of the small step.

response to an input current. First, it gives us the voltage where we expect V_s to settle:

$$V_s = U_T \ln \frac{I_{in}}{I_{bias}}. \tag{27}$$

Second, the picture tells us that we will get small time constants for large values of I_{in}. Note from Eq. 26 that the vertical offset of this plot is determined by the value of I_{in}. As I_{in} increases, the plot is shifted up, and the rate at which V_s changes for a given value of V_s will be increased, thus decreasing the time constant. It is also important to point out that the slope of the actual phase portrait is much steeper than what we drew in Fig. 19(b). This means that a shift up in the plot won't affect the steady-state value of V_s as much as it will affect the time constant.

3.4.2 Demonstration of Impact on Dendrite Circuit Behavior

If we apply a large enough input current such that the membrane voltage changes by more than 100 mV, we can measure the effects of nonlinear input currents on the dendrite.

Our first experiment was to observe the steady-state voltage decays, as shown in Fig.

20a. The result is that the voltage decays linearly with space. This is a desirable effect, since it is essentially a compression operation. For small inputs, the steady-state voltage decayed exponentially. If this trend were to continue for large inputs, the dynamic range of available voltages would be severely limited. However, for a large input, the FETs are no longer operating as resistors; they are in saturation, so we merely require linear changes in voltage to achieve exponential changes in the leakage current. Therefore the dendrite is using nonlinearity to increase its dynamic range.

Our second experiment was to observe how the shape of the step response changes with an increase in input current. We can rewrite Eq. 28 in the current domain. Defining $I_1 = I_{bias} e^{V_s/U_T}$, we can differentiate with respect to time to get $\dot{I}_1 = I_1/U_T \dot{V}_s$. Substituting into Eq. 26,

$$\begin{aligned} I_{in} &= \frac{CU_T}{I_1}\frac{\partial I_1}{\partial t} + I_1 \\ \frac{\partial I_1}{\partial t} &= \frac{I_1}{CU_T}\left(I_{in} - I_1\right). \end{aligned} \tag{28}$$

When Eq. 28 is solved, it behaves like a tanh function, so we expect the shape of our dendrite's step response to be sigmoidal for large current steps. Our results in Fig. 20 bear this out.

3.5 Implementing Dendrites in Large Reconfigurable Systems

The FPAA connects analog components together using a "switch matrix" which consists of a dense array of floating-gate pFETs. A benefit of using floating-gate pFETs as the switching elements is that we are not restricted to use them only as switches. Since they are analog circuit elements, they can be used as such. For instance, they can be used as current sources or cascodes. When working in a neuromorphic application, it is possible to connect the switch matrix FGs together to implement dendritic cable sections. Rather than using a DAC to explicitly apply a gate voltage to the horizontal and vertical transistors, we can program the floating-gate pFETs with varying levels of charge. The switch matrix is an extremely dense array of switches, so we can make large dendrites as inputs into neurons.

3.5.1 Difficulties of Floating-Gate Diffusors

Modeling floating-gate denditic circuits is more complicated than with regular FETs. The capacitive coupling from the source and drain to the floating-gate is more pronounced than with regular pFETs. To design a floating-gate dendrite, an extra step of characterizating these coupling ratios is necessary. If we desire more complicated behavior by programming different values of the floating-gate voltage for different sections of the dendrite (i.e. changing the dendrite's diameter), we will need to take these coupling ratios into account when determining to what voltage we want to program the floating gate. We need to know coupling ratios because floating-gate transistors are programmed with their terminal voltages at one potential (in "program mode"), and after programming their terminal potentials undergo a change (in "run mode," when the circuit is operating). An example of a floating-gate diffusor not behaving as expected is shown in Fig. 21.

The simplest way to characterize the capacitive coupling is to perform sweeps of each terminal and extract an "effective κ" for that terminal. This measure tells how much a change in one terminal voltage will modify the floating-gate potential. Then if we have a desired membrane potential, we know how much all of the terminals will change in the transition from program mode to run mode, and we know how the floating-gate voltage will be affected. Once we know that floating-gate voltage, we can attempt to program the bias transistor to match the current it is drawing. More than likely, we will require an iterative process of programming the bias transistors, measuring the membrane voltage, and reprogramming to achieve the desired membrane potential.

Another important nonideality in floating-gate systems which requires characterization is the that the transistor which is programmed differs from the transistor which is actually placed in the circuit. This scheme is known as indirect programming, and any differences between the programmed and in-circuit transistor will affect the circuit's performance. Methods to characterize these effects are discussed in [56].

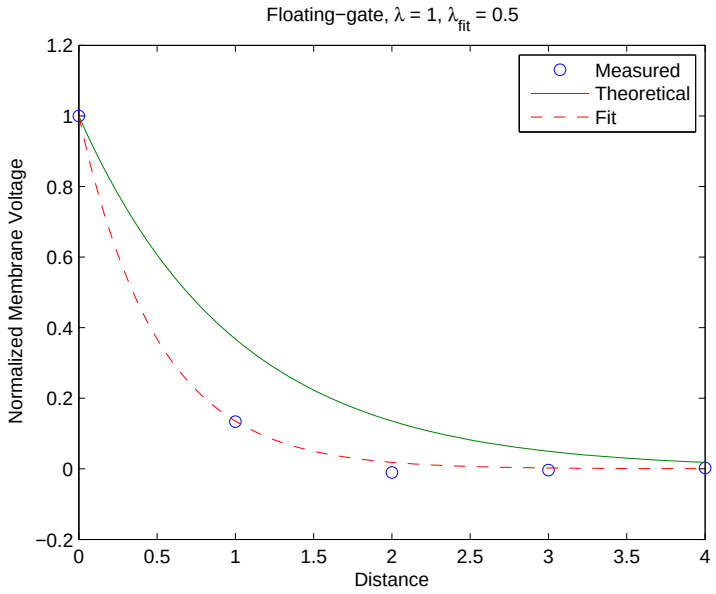


Figure 21: Illustration of offsets introduced by capacitive coupling from the drain of the diffusor.

3.5.2 Benefits of Floating-Gate Diffusors

The most exciting aspect of dendritic circuits is that they can be made in an extremely compact manner. As we stated above, the switch matrix of the RASP 2.8a FPAA is completely made up of floating-gate switches. So there is potential to make huge arrays of dendrites using the switch matrix. Since the purpose of the array is to interconnect components, it makes sense that dendrites be used to send signals from one compiled structure to another. Figure 22 is an example of how such a diffusor might be made. Partitioning of the switch matrix allows for a large number of dendrites to be created.

We can estimate how large these dendrites can be based on the FPAA routing structure. Each CAB has an associated floating-gate switch matrix. Some rows and columns are global, meaning they have connectivity among multiple CABs. We will only consider local rows and columns which do not connect beyond a CAB. In addition, the columns have semi-local connections to their nearest vertical and horizontal neighbors, so we assume that half of those columns are available per CAB. The equivalent number of useful columns per CAB is 14.The rows are hard-wired to CAB elements, so the number of usable rows is reduced to ensure no CAB devices are turned on. For CAB types 1 and 2, the number of available rows

is 24 and 34, respectively. If we make a dendrite as shown in Figure 22, each row connects to one vertical transistor, and each column connects to two horizontal transistors. The size of dendrites in CAB type 1 is limited by its number of rows, while CAB type 2 is limited by columns. Therefore, we estimate that CAB types 1 and 2 can implement dendrites of approximately 24 and 28 stages, respectively. Based on the numbers of these CABs in the FPAA, we can theoretically make 28 dendrites of length 24 and 4 dendrites of length 28. We can then use the global routing to chain some of these together.

Neural systems are inherently imprecise. Real synapses are unreliable, and no two dendritic structures are the same. So the disadvantages listed above are not necessarily detriments. Some amount of variability from dendrite to dendrite caused by floating-gate transistor mismatch could be a good thing. The inability to precisely model the behavior could be an asset, for it requires designers to get an intuitive feel for what parameters work well for a given system.

3.6 Conclusion

We have demonstrated that it is possible to use an FPAA to create a voltage-mode CMOS dendrite which maintains certain properties of linear cables. We have seen qualitative behaviors that are similar in both the steady-state response and the dynamic response. With this as a fundamental building block in neuromorphic circuits, we are now free to explore more interesting topologies.

The next step in this research is to demonstrate computational primitives using these dendritic structures. Simple dendritic computations have been proposed for a long time, such as coincidence detection and simple boolean operations caused by local inhibition [32], [40]. These computations are often supported by active channels [60]. For example, a recent paper showed that both the passive properties of linear cables and the nonlinear effects of NMDA channels cause dendrites to respond more strongly to a centripetal sequence of inputs than a centrifugal sequence [9], [15]. These simple computations have the potential to form more powerful units. Specifically, we have proposed that they could be used to aid HMM-like classification [25], and we are working towards demonstrating this functionality

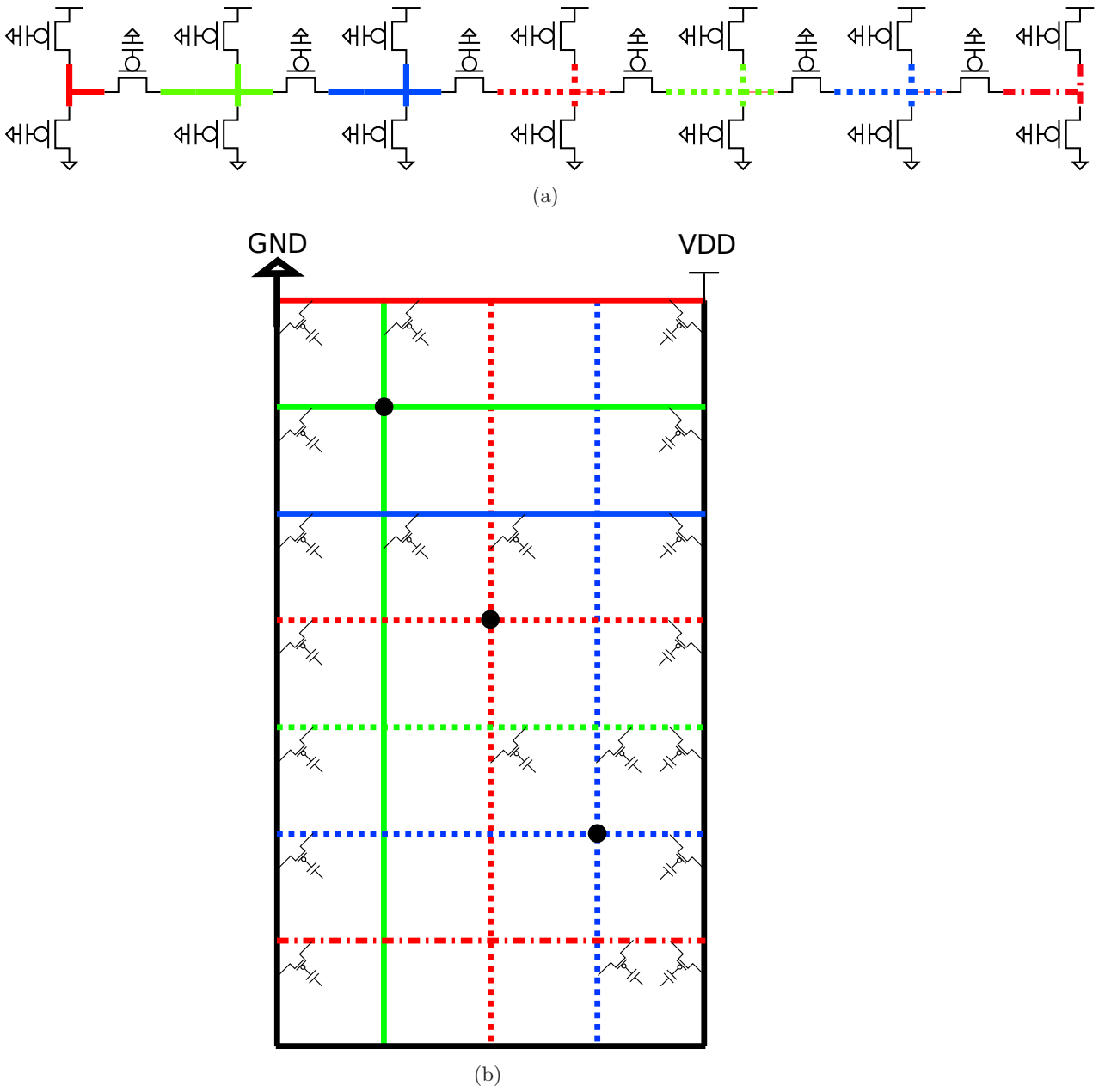


Figure 22: Possible method of placing dendrite in switch matrix. (a) Schematic of diffuser with nets colored and dotted to represent different nodes. (b) In a switch matrix, a floating-gate transistor exists at every intersection of two wires which can short a horizontal line and vertical line. An intersection with a black dot represents wires which have been shorted together with a floating-gate transistor. A picture of a transistor represents a floating-gate which is part of the diffusor structure and is programmed somewhere between open and closed circuit. No graphic at an intersection represents a floating-gate which has been programmed open-circuit. The leftmost column has been shorted to ground, and all the transistors connected to it are the vertical devices in the diffusor. The rightmost column has been shorted to V_{DD}, and all the transistors connected to it are the biasing devices. The pairs of two floating-gates in the middle are the horizontal transistors which connect the vertical legs together.

experimentally. We hope to leverage these units for useful classification and discrimination systems.

CHAPTER IV

SUBTRACTIVE AUDIO SYNTHESIS ON A FIELD-PROGRAMMABLE ANALOG ARRAY

This chapter discusses the implementation of a subtractive audio synthesizer on the FPAA. We added new infrastructure to our toolset which helped create consistent results. This new feature is the ability to specify the placement of CAB components at the Simulink level, which allows users to reuse circuits they have already characterized. We also integrated characterization into the high-level Simulink structure. The result was a system that enabled more repeatable and predictable results.

4.1 Reconfigurability Enables Integrated Circuits for Analog Music Synthesis

One issue preventing analog music synthesis systems from being integrated onto a single chip is that most analog ICs are not reconfigurable. There is oftentimes a cycle of design, testing, and redesign that is time- and money-intensive [63]. This is an especially difficult problem with music synthesis systems, because sonic quality is more subjective than other applications. A reasonable alternative is to use a reconfigurable analog chip which can allow designers to tweak parameters until the desired sonic quality is achieved. In this chapter we present the basics of subtractive analog synthesis and how our reconfigurable platform operates.

Subtractive synthesis is the process of creating sound by beginning with a harmonically rich signal and then removing elements of its amplitude and frequency – "subtracting" from it, in a sense. Fig. 4.1 shows a block diagram of a subtractive synthesis system [2],[29]. The first block is a voltage controlled oscillator, or VCO. The frequency of the waveform is controlled by one or many control voltages, or CVs. This generates a periodic waveform which forms the basis of the sound.

The next circuit in the chain is the voltage-controlled amplifier, or VCA. The gain

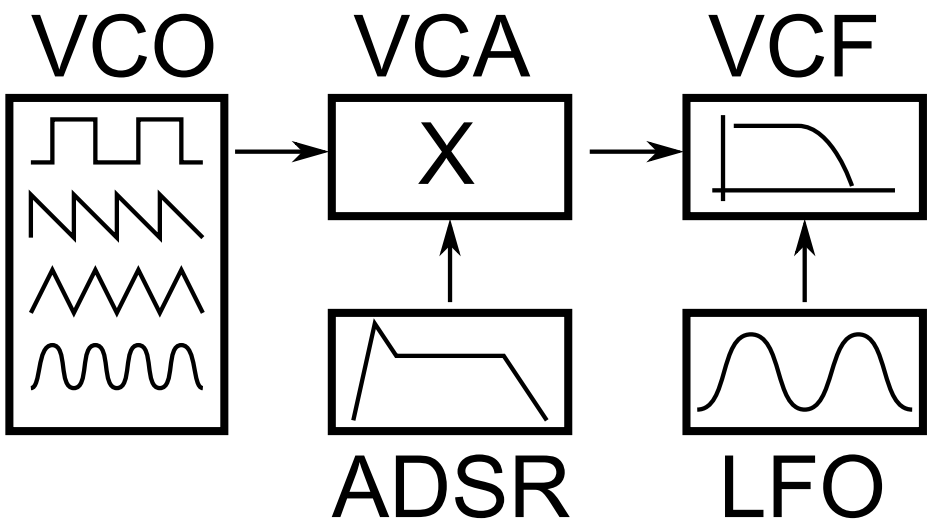


Figure 23: Basic blocks of subtractive synthesis. A voltage-controlled oscillator generates the raw signal. A voltage-controlled amplifier provides the envelope of the signal, typically of the attack-decay-sustain-release shape. The waveform is then passed into a voltage-controlled filter. The cutoff frequency is set by a control voltage. The cutoff frequency is often modified by a low-frequency sinusoid, known as a low-frequency oscillator.

is controlled by a CV. This will provide the amplitude envelope of the signal. Musical instruments have different amplitude dynamics, and the VCA enables emulation of many different instruments. A popular envelope in subtractive synthesis is known as ADSR, or attack-decay-sustain-release. Its envelope rises to its maximum amplitude quickly (attack), decays to a lower amplitude (decay), remains constant for a period of time (sustain), and then falls back to zero (release).

Finally, the waveform is passed into a voltage-controlled filter (VCF). The cutoff frequency is determined by a CV. There are numerous forms of these filters, but popular topologies generally include high orders, nonlinearity, and resonance.

Oftentimes the CV for a VCF is not a DC voltage but a slowly varying sinusoid that comes from a low frequency oscillator (LFO). The LFO causes the cutoff frequency to vary slowly, generating a "wah" type sound.

4.2 Extensions of High-Level FPAA Tools

Our lab has developed software tools that make the design of circuits on the FPAA easier. Programming of the FPAA is controlled through a suite of Matlab software which interfaces

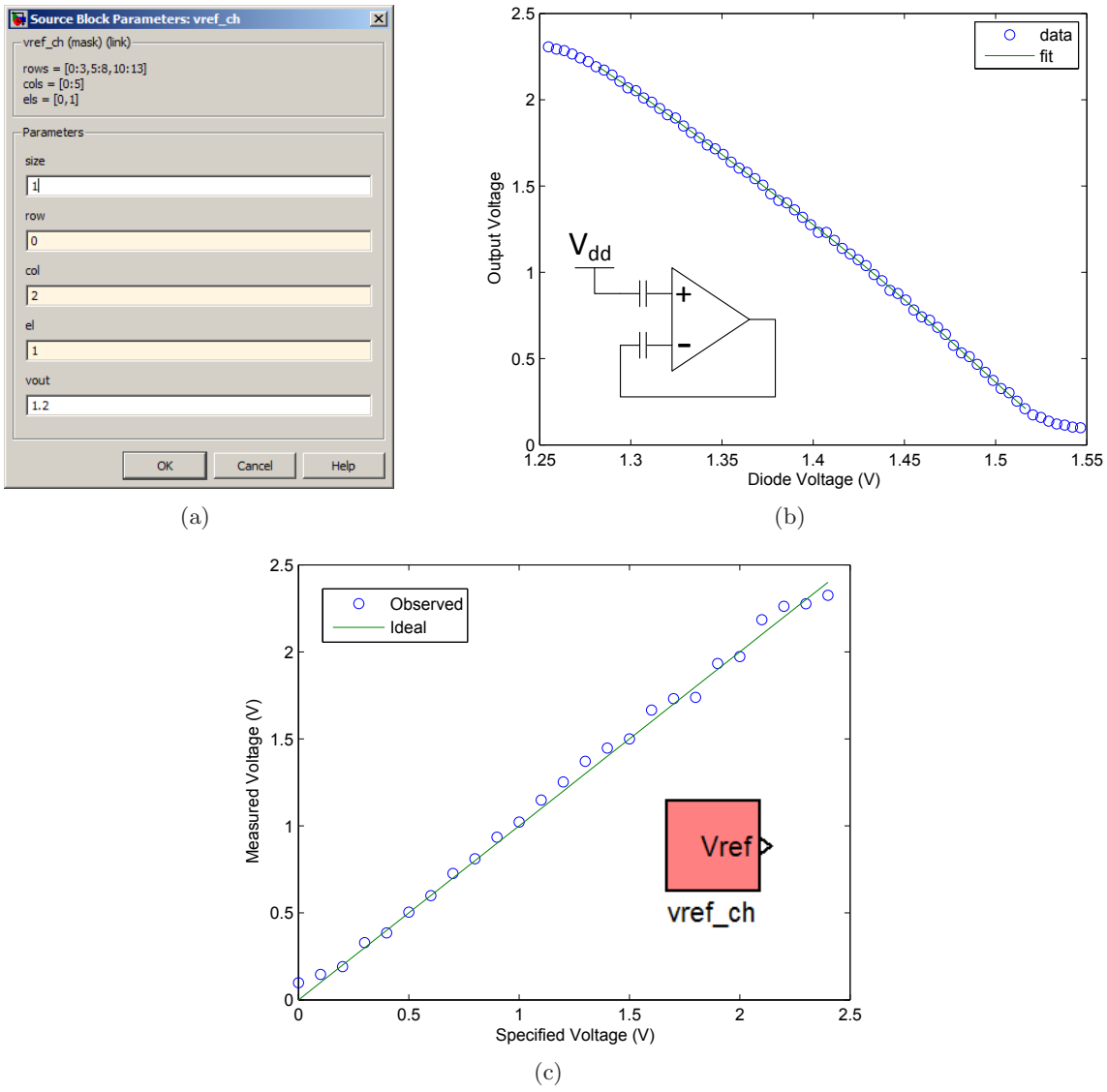


Figure 24: New Simulink tools were developed during this project. (a) Interface to a voltage reference block. The boxes labeled "row," "col," and "el" specify the address where this this reference element should be force placed. The box labeled "vref" shows that users don't have to directly enter floating-gate targets. They enter the desired reference voltage, and the targets are picked automatically. (b) The circuit (see inset) is a floating-gate input OTA in negative feedback with one terminal at V_{dd}. The plot is an example of how we characterize the circuit. One of the FGOTA's terminals is swept over a wide floating-gate range (each V_{fg} corresponds to a diode voltage), and the output voltage is measured. A polynomial is then fit to the characteristic and stored. (c) Sweeping the vref parameter of the circuit results in a linear reference. The accuracy can be improved by reducing programming errors. We used this circuit for voltage references that needed an accuracy of around 50 mV.

to an Atmel microcontroller. We have developed a Simulink front-end to the programming scripts called Sim2Spice [51]. Users create a Simulink block diagram containing components of the FPAA connected together to create the desired circuit. Sim2Spice then parses the diagram and generates a netlist that the programming scripts can then target to the FPAA.

One advantage of Sim2Spice is that users can create their own Simulink blocks which represent higher-level collections of circuits. As a result, block diagrams that were once complicated can be greatly simplified. They can also be paramaterized so that users can change aspects of the circuit from the higher-level interface. For this project, we have created a library of Simulink blocks for some of the most-used circuits in audio synthesis, and throughout the chapter we will show the blocks and how they connect together to create synthesizer subsystems.

We found that some extensions to the Simulink toolset would help us accomplish our goals more easily, so new tools were created in the course of this project.

4.2.1 Force Placement via Simulink

Formerly, the software flow did not allow users to force the placement of their components into particular CAB locations on the FPAA from the Simulink interface. They would have to edit the netlists by hand to get this functionality. It is clearly better to force the placement of components from Simulink to ensure that any characterization of a particular circuit will remain constant throughout multiple program cycles. Once a user characterizes a particular circuit, he can force that circuit to always be placed in the same location and never worry about re-characterizing.

An example of a Simulink block which allows force placement is shown in Fig. 24(a). Users enter information which designates where a block should be physically placed in the FPAA, and the corresponding device will be used.

4.2.2 Integrating Characterization Into Simulink Tools

Another requirement for this project was to enable the software to track characterization data of FPAA components and automatically use this data when a user places a block. This makes it easier for users to simply use blocks without worrying about their characterization.

An example of this is creating a voltage reference. The schematic for a voltage reference is shown in the inset of Fig. 24(b). It is a floating-gate OTA in negative feedback with the positive terminal at V_{dd}. Users can program various amounts of offset between the two input terminals to make the output voltage vary. The inset of Fig. 24(c) shows the Simulink block created for this circuit. The problem with the top schematic is that users don't want to have to characterize what input offset voltages result in their desired output voltage. They would much rather specify their desired output voltage and have the program decide what input offsets to use. We have created a new block to solve this problem.

Fig. 24(b) shows the characterization experiment that is run for every FGOTA reference used in our system. The negative terminal's floating-gate voltage is set to V_{dd}, and the positive terminal's value is swept. A polynomial is then fit to the characteristic of this device. Once this characterization has been performed, the user can use the voltage reference block, which has a user interface as shown in Fig. 24(a). The steps the software goes through can be summarized as follows:

Characterization

1. Experimentally determine $I_{target,p}$ vs. V_{out}
2. Fit polynomial to the plot
3. Save polynomial to *.mat file

Compilation

1. User places vref block with a desired output voltage
2. Software looks up and evaluates characterization polynomial
3. Software places correct current target into netlist
4. Voltage reference is programmed

4.3 Voltage-Controlled Oscillator (VCO)

The first step in audio synthesis is to create a periodic waveform. The frequency of the waveform should be tunable over a wide range, and characteristics of the waveform should

be controllable by CVs. Our VCO is based on a hysteretic comparator and current-starved inverters.

4.3.1 Current-Starved Inverter

A simple subcircuit we use frequently is the current-starved inverter (CSI). A CSI is an inverter that is slew-rate limited by design, as shown in Fig. 25(a). This limitation is generated by a pFET and nFET cascode which can only source or sink a small current I_{lim} to the capacitance C_{out} at the output node. The voltage slews at the rate $\partial V/\partial t = I_{lim}/C_{out}$. We symbolize the CSI in future schematics using the symbol in Fig. 25(b), and its Simulink block is shown in Fig. 25(c). When this circuit is compiled, we get results as shown in Fig. 25(d).

A current-starved inverter is a useful tool because it translates square-wave inputs into triangle waves. This will be used in a number of waveshaping circuits in the following sections.

4.3.2 A VCO based on CSIs

Fig. 26(a) shows the Voltage-Controlled Oscillator (VCO), which has an associated Simulink block as shown in Fig. 26(b). The behavior of this circuit is shown in the timing diagram in Fig. 26(c). We start by assuming that node V_{sqr} is already oscillating. It is at the output of an OTA with a high bias current, so we can assume it is a square wave. This signal enters a current-starved inverter to produce V_{tri}. So the slope of the output triangle waveform depends on the pFET and nFET slewing currents in the triangle CSI, denoted as $I_{Tri_{n,p}}$.

Next, V_{tri} is fed through two inverters to generate a square wave, V_i, which is high when V_{tri} is above V_{ref}. So now we have a current summing onto the capacitor C_1: I_{cmp}. That current determines how quickly V_{cmp} charges and discharges, and therefore the timing of when V_{sqr} goes high or low.

In summary, $I_{Tri_{n,p}}$ determine the slopes of the triangle waveform, and $I_{cmp_{n,p}}$ determine when V_{tri} changes from slewing up to slewing down. This can be translated into frequency and amplitude of the triangle waveform, and we have control over the frequency of the square wave.

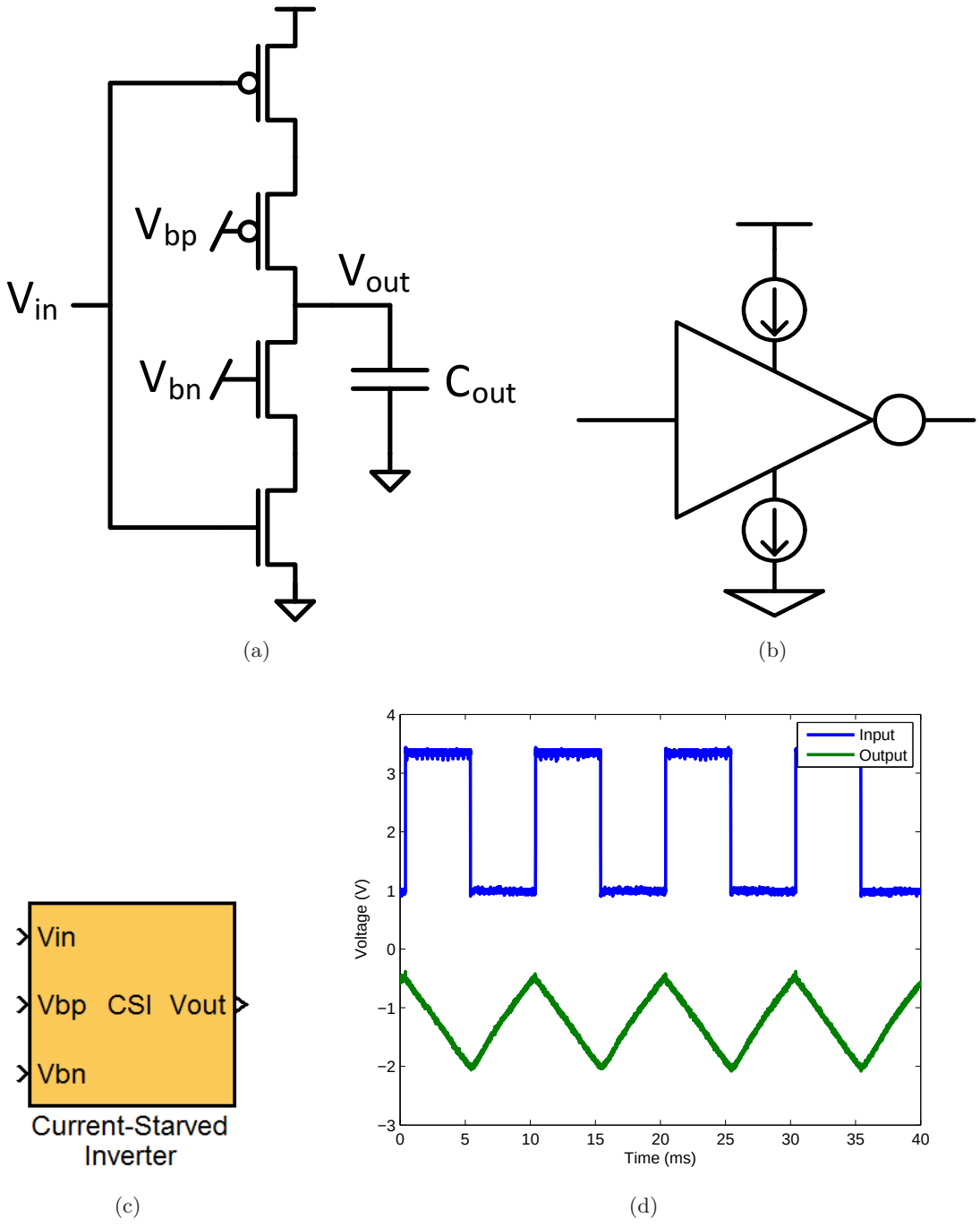


Figure 25: Demonstration of the design and operation of a Current-Starved Inverter (CSI) (a) Schematic of a CSI. (b) Symbol for CSI. This will be used to simplify schematics of future blocks. (c) Simulink block created for CSI. (c) Data from a CSI. Choosing correct values for the limiting current will result in a triangle waveform being generated from a square input. DC offsets removed for illustrative purposes.

Finally, the triangle is converted into a sinusoid by sending it into an OTA [61]. OTAs have a saturating nonlinearity in the form of a hyperbolic tangent function. So when a

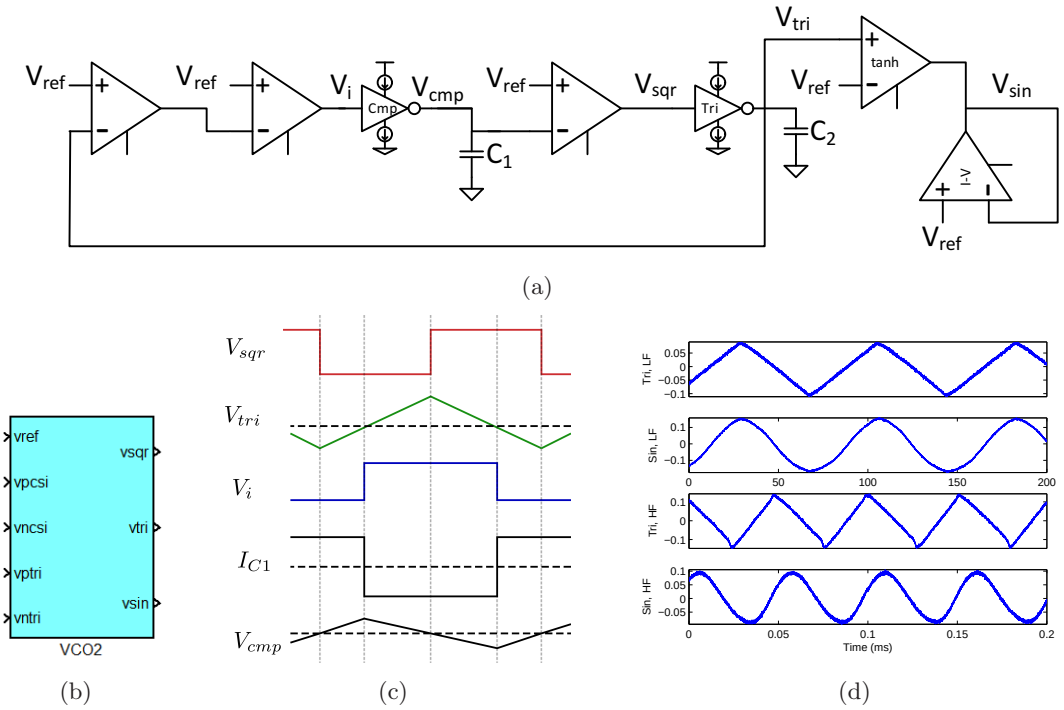


Figure 26: The voltage-controlled oscillator forms the basis of the sound in a subtractive synth. (a) Schematic of the oscillator circuit. We make use of the Current-Starved Inverter to shape digital pulses into triangle waves, which are then compared to a reference voltage. The output of this comparator generates a square wave, and the triangle wave is seen at the output of the CSI. We then use an OTA as a tanh function and current-to-voltage converter to turn the square wave into an approximate sine wave. (b) Simulink block of VCO. The reference voltage and biases for the CSIs are inputs to the block. (c) Timing diagram of various signals in the oscillator circuit. We essentially have two square wave and triangle waves out of phase with each other, which causes oscillation in the circuit. (d) Output of triangle and sine wave generation subcircuits for high and low frequencies. The period of the low and high frequency waveforms are 77.2 and 0.0515 ms, respectively. The square wave is not shown because pinning it out for this particular compile caused it to couple to the other signals. For this and all other waveforms in the chapter, unless otherwise noted, DC offsets are not shown for illustrative purposes.

triangle with the correct amplitude is fed into the OTA, the output becomes sinusoidal in shape. The follower-connected OTA labeled $I - V$ converts the output current into a voltage, so we get a sinusoidal waveform at the output.

Fig. 26(d) shows the triangle and sinusoidal waveforms at different frequencies.

4.4 Voltage-Controlled Amplifier (VCA)

The VCA controls the amplitude envelope of the output from the VCO. Pure triangle or square waves are not musically interesting until their amplitudes change in time. The inputs

to the VCA are the desired envelope and the constant-amplitude waveform, and the VCA multiplies these two to create a sound that mimics an instrument's amplitude profile. We use a Gilbert Multiplier with a current mirror output stage to create the VCA. We also introduce a method to create a popular envelope known as the attack-decay-sustain-release envelope.

4.4.1 The Gilbert Multiplier

One CAB element on the FPAA is particularly well-suited to act as a VCA. The Gilbert Multiplier is a well-known signal processing element. As shown in Fig. 27(a), it is a five-terminal device with two differential input voltages and one output current. The VCA is made by connecting a current mirror to the output to turn the output current into a voltage, and that combination is placed into the Simulink block shown in Fig. 27(b). From [43], the equation describing the circuit's operation is

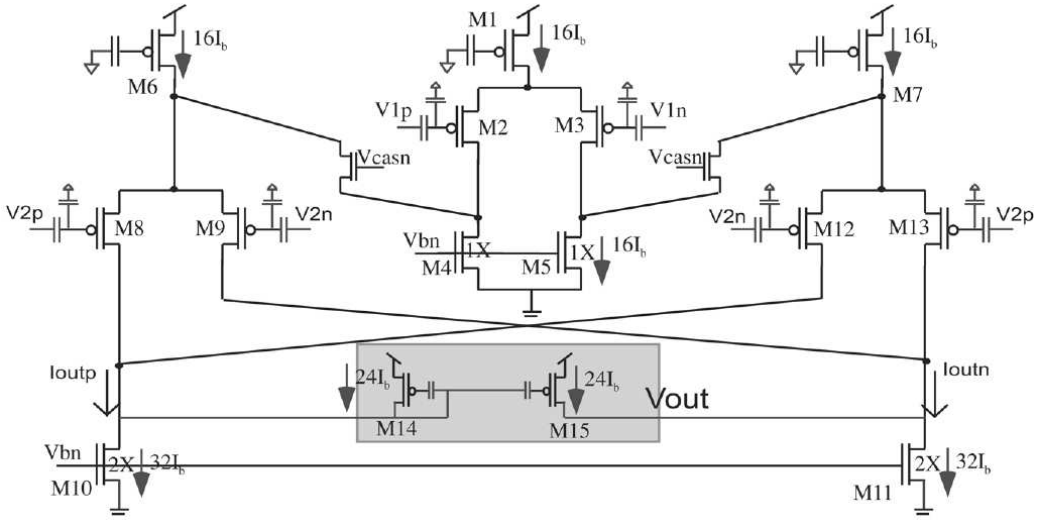
$$I_{out} = I_b \tanh \frac{\kappa(V_1 - V_2)}{2U_T} \tanh \frac{\kappa(V_3 - V_4)}{2U_T}. \tag{29}$$

A DC sweep of the input is shown Fig. 27(c).

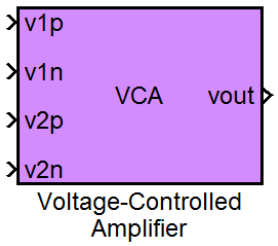
4.4.2 Removing DC Offsets

One problem with the Gilbert multiplier is input and output DC offsets. When the circuit is used in an audio context, input offsets are translated into gain errors or distortion. However, since the input pairs of the circuit are floating-gate pFETs, we can remove a significant portion of the DC offset. An example of a method for dealing with this offset is to change the floating-gate voltages of two transistors in the circuit: one of the input transistors, and one of the output transistors. We divided the offset removal into two steps. First, we removed the output offset so that the output signal could be seen clearly. Next, we removed the input offset. We then repeated this process to fine-tune the results.

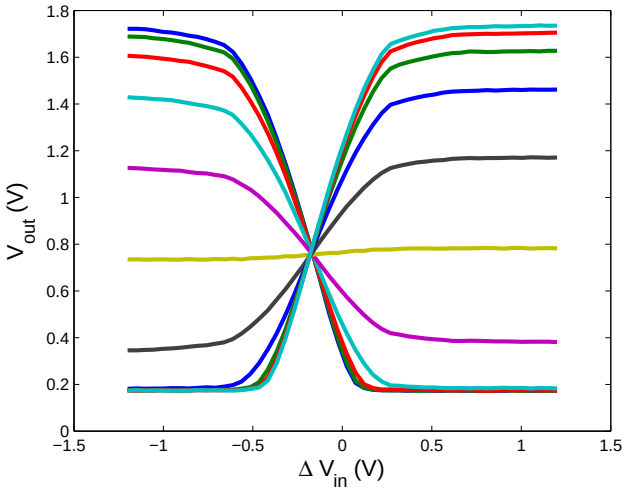
The insets of Fig. 28(a) show the difference between no offset removal and complete offset removal. The horizontal line demonstrates how much output offset has been removed. The blue line is the characterization of output offset. In this experiment, we measured the value of the output voltage when $V_{id} = 0$ for different values of floating-gate charge in the



(a)



(b)



(c)

Figure 27: The Gilbert Multiplier is a voltage-controlled amplifier. (a) Schematic of the Gilbert Multiplier (image from [4]). (b) Simulink block combining the Gilbert Multiplier and current-mirror output stage. (c) DC Sweep of $V_1 - V_2$ for different values of $V_3 - V_4$.

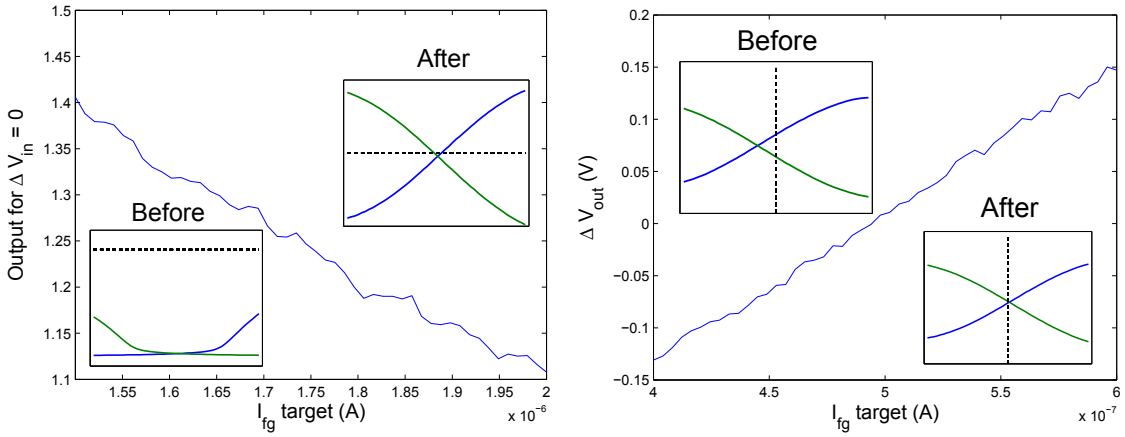


Figure 28: Removal of offsets in the voltage-controlled amplifier. (a) Characterization of output offset and demonstration of the results of output offset removal. Before offset has been removed, the outputs at max and min gain are much below midrail (1.2 V). After both offsets have been removed, the output is approximately 1.2V. (b) Removal of input offset. Before the input offset has been removed, the outputs at max and min gain do not intersect when $V_{id} = 0$. After both input and output offsets have been removed, they do intersect at approximately $V_{id} = 0$.

output current mirror. The floating-gate charge is represented by the amount of current flowing through the FET (I_{fg} target) for fixed terminal voltages. We were able to achieve a common-mode output voltage of approximately 1.2 V after offset removal.

Fig. 28(b) demonstrates how input offset removal shifts the curve closer to the origin. We programmed different values for the initial charge at the gate of one of the input devices, while leaving the other terminal at a constant initial charge. After each programming cycle, we swept the differential input voltage for the highest possible gain and the lowest possible gain (i.e. for $V_{2p} - V_{2n} = 2.4$ and $V_{2p} - V_{2n} = -2.4$). We then took the difference between the output voltages when $V_{1p} = V_{2p}$. There is no input offset when this difference is 0V.

4.4.3 The Attack-Decay-Sustain-Release (ADSR) Envelope

Attack decay sustain release (ADSR) is a waveform envelope used frequently in subtractive synthesis. When this envelope is applied to a waveform of constant amplitude, the result sounds much like a plucked-string instrument. An ADSR waveform is created by a piecewise-linear function, schematically illustrated in Fig. 4.1. The FPAA is well-suited to generating such waveforms using CSIs.

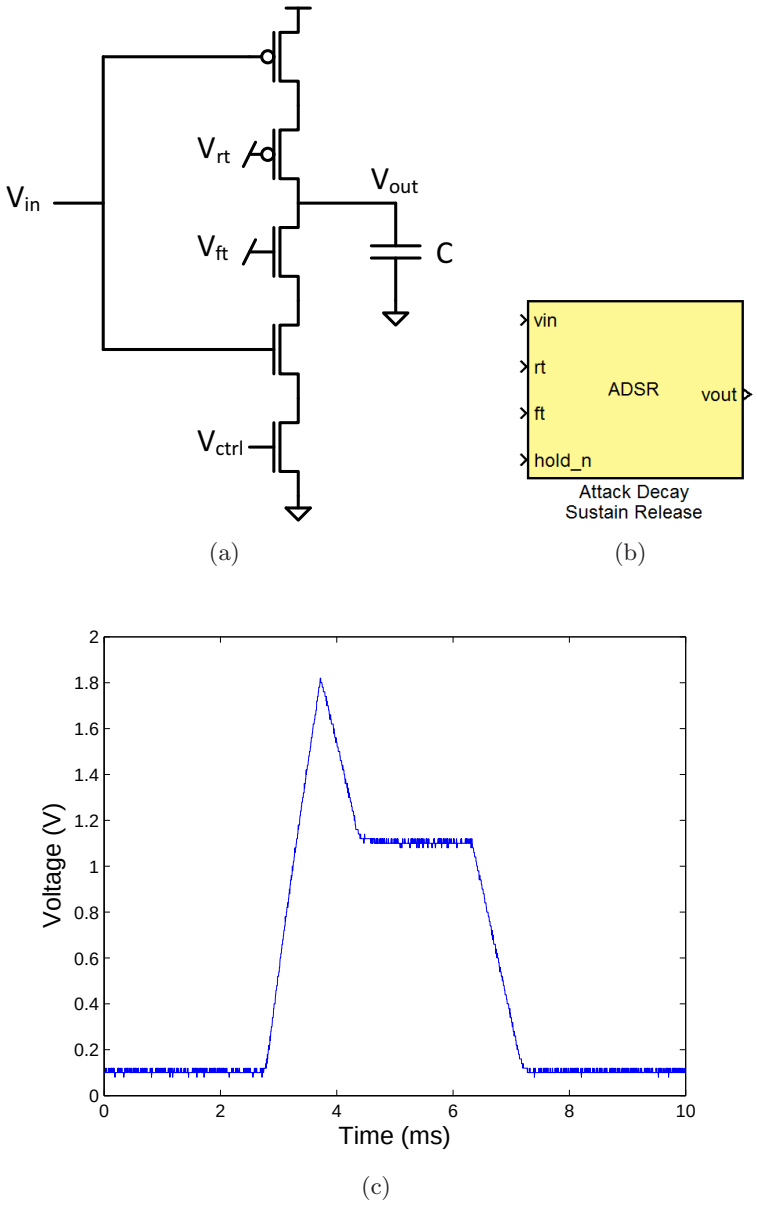


Figure 29: The attack-decay-sustain-release envelope is applied to a waveform to make it sound like a plucked-string instrument. (a) Schematic of the ADSR circuit. This is a CSI with a cutoff switch to allow for a sustain stage. (b) Simulink block for ADSR envelope creation. (c) Waveform measured from the ADSR circuit. One 500fF capacitor was placed at the output node.

The ADSR waveform consists of slewing up and down, holding at a constant potential, and finally slewing down. We can create this shape by adding one element to the CSI: a cutoff transistor to make $I_{lim} = 0$ during the Sustain stage, as shown in Fig. 29(a). Since no current charges the cap, the voltage should hold constant assuming small leakage currents.

Increasing the capacitance at the output node should make the circuit leak more slowly. The Simulink block developed for the library is shown in Fig. 29(b). A measured waveform generated by this circuit is shown in Fig. 29(c).

4.5 Voltage-Controlled Filter (VCF)

A conceptual schematic of our FPAA implementation of the transistor ladder filter is shown in Fig. 30, and its Simulink implementation is shown in Fig. 31. The original transister ladder used NPN transistors, which are analagous to nFETs. However, we have chosen to use pFETs because they are present in floating-gate form on our target FPAA, and we want the option to implement the ladder using these devices. The bias voltages V_{b1} through V_{b4} at the gates of the MOSFETs are ideal candidates for a floating-gate configuration, since after an initial calibration procedure these bias voltages would generally remain fixed. [1] The original transistor ladder filter exploited the dynamic resistance of BJTs; we similarly exploit the dynamic resistance of subthreshold MOSFETs, with the V_{cutoff} controlling the bias current produced by the topmost transistor in Fig. 30. This bias current varies the dynamic resistance of the MOSFETs and hence the effective cutoff frequency of the filter stages, just as in the original BJT-based filter. Most implementations of the transistor ladder filter, both commercial and hobbyist, have used NPN transistors; implementations using PNP transistors exist, but are relatively rare. Hence, our ladder schematic may look "upside down" relative to the transistor ladder schematics most enthusiasts are accustomed to seeing.

We used a simple two-OTA attenuator at the input to reduce the audio signal to a level such that the filter acts approximately linear.[2] We used a similar two-OTA amplifier at the output to compensate for this initial attenuation.

The transfer function for this topology has been derived in detail elsewhere [58]. In that analysis, BJTs were used rather than MOSFETs. Fortunately, MOSFETs acting in the

[1] For ease of prototyping, the implementation shown in this paper use CAB pFETs that have gates driven by DACs on our development board; these could be readily changed to floating-gate pFETs with an appropriate programming procedure; we leave this as an avenue for future work.

[2] Pushing the input level to a point that the filter slightly distorts the signal is not necessarily "bad," as subtle nonlinear effects are commonly thought to contribute to the "fatness" of the transistor ladder filter sound.

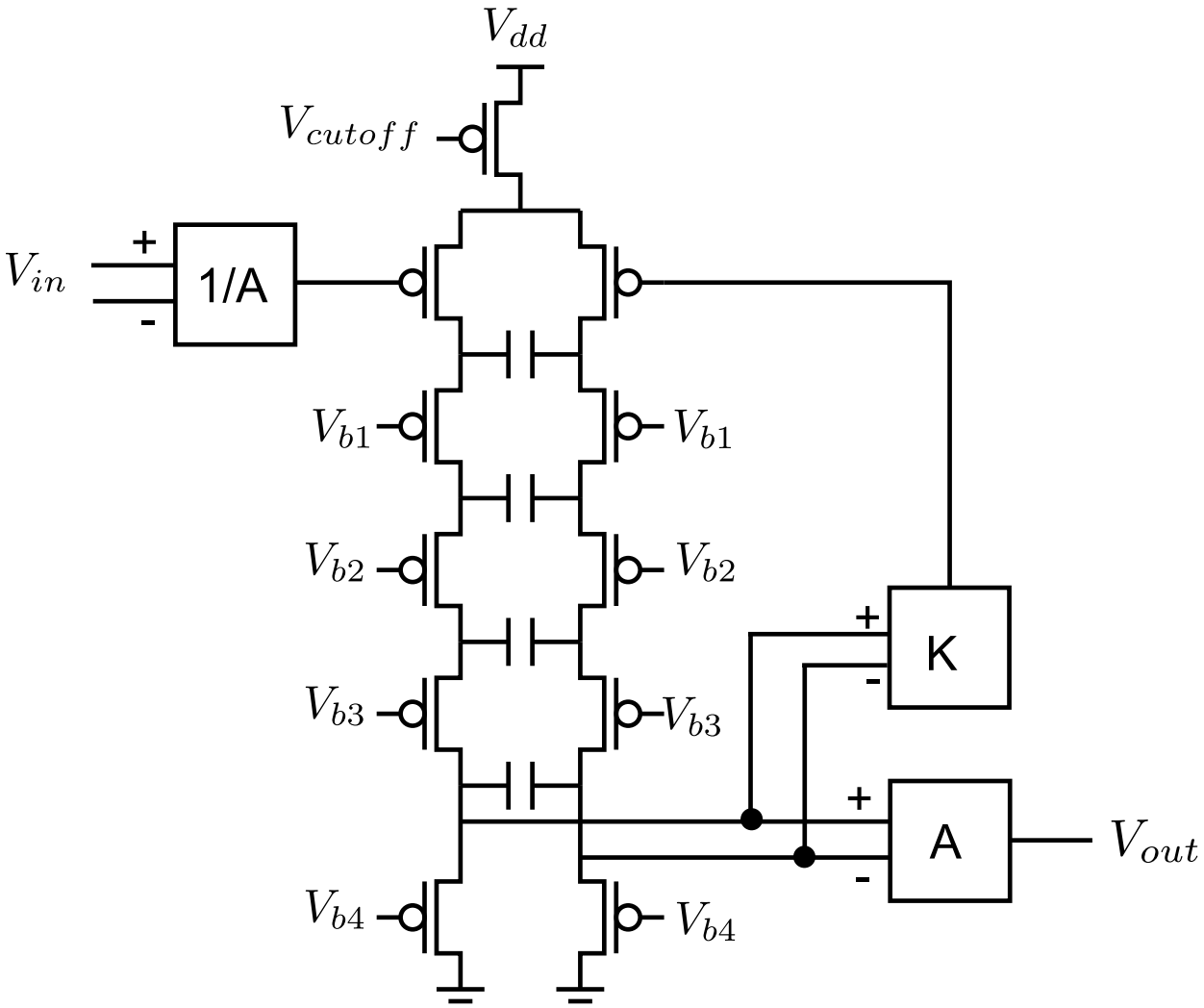


Figure 30: Conceptual schematic of an implementation of ladder filter tailored for pFETs operating in the subthreshold regime. An input is fed into the left side of the filter, and it passes through a four filter stages. A differential output is taken between the left and right side of the ladder with a negative feedback gain of K. In our implementation, feedback was achieved using a two-OTA voltage amplifier. We also attenuated the signal before applying to the input to avoid distortion, and then amplified the output.

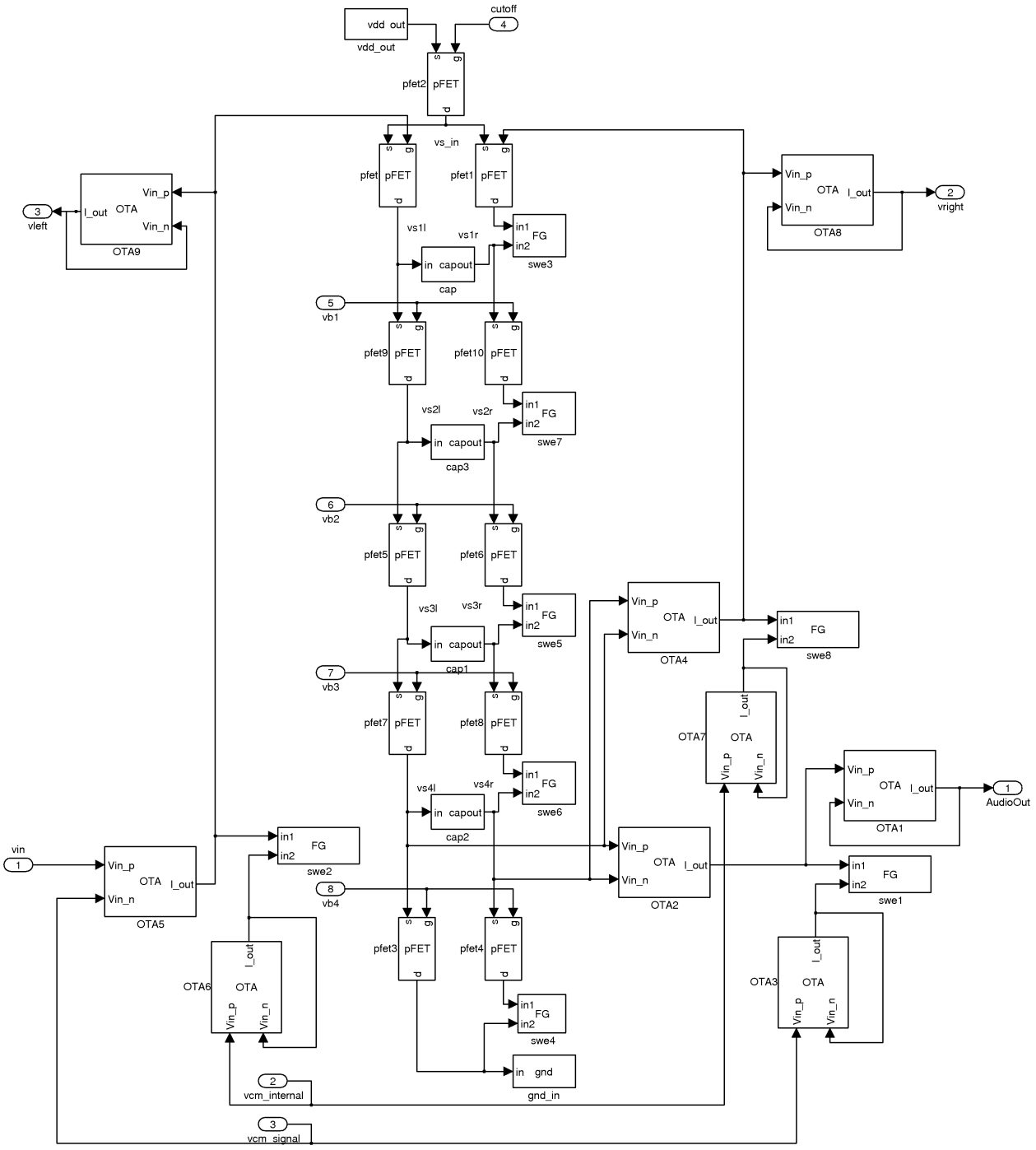


Figure 31: Simulink model used to take the data presented in this paper. CAB pFETs and OTAs are used for this implementation. For the "k" amplifier, the differential V-I OTA's bias current was swept between 250 nA and 4 μA. The I-V OTA's bias current was left at 1μA. The "A" and "1/A" OTAs had biases of 200 nA and 1 μA, depending on whether the gain was more or less than one. The output buffers are biased at 5 or 10 μA. Ladder capacitance values are approximately 2 pF. The "FG" elements are effectively short-circuits, used to circumvent Simulink's disallowing block outputs to be connected together. Physically they are compiled to fully-on floating-gate routing elements.

subthreshold [3] regime have exponential I-V characteristics, similar to BJTs. For a pFET in saturation with terminals referenced to the bulk, and if drain effects are ignored, this characteristic simplifies to

$$I = I_0 e^{(V_s - \kappa V_g)/U_T}, \tag{30}$$

where I_0 is a grouping of physical constants, V_s is the source voltage, V_g is the gate voltage, κ is a physical constant representing the gate to channel coupling, and U_T is the thermal voltage. Typical values for κ are around 0.7.

The transfer function is developed as in [58] by small-signal analysis of these devices. The input pair converts an input voltage into current through its transconductance

$$g_m = \frac{\partial I}{\partial V_g} = -\frac{\kappa}{U_T} I, \tag{31}$$

where I is the bias current. The ladder's gates are at fixed potentials, but their source voltages are changing as the input changes, so the source conductance is modeled as

$$g_s = \frac{\partial I}{\partial V_s} = \frac{1}{U_T} I. \tag{32}$$

Developing as in [58], the closed-loop transfer function is

$$H(s) = \frac{\kappa}{\left(\frac{2sC}{g_s} + 1\right)^4 + k\kappa}. \tag{33}$$

We have now demonstrated that changing the bias current will modulate g_s and therefore change the cutoff frequency.

The k factor determines the amount of negative feedback from the output of the four stages back to the input. Each stage approximately buffers the next, so when $k = 0$, the response is that of a cascade of four identical single-pole sections. Each section provides a 45 degree phase shift at its half-power cutoff frequency, for a total phase shift of 180 degrees. Moog's insight [45] was that negative feedback could result in a musically useful enhancement around the cutoff frequency, with the DC gain decreasing with increasing k.

[3] MOSFETs are most commonly thought of as having square-law, instead of exponential, I-V characteristics. Unfortunately, some undergraduate electronics texts mischaracterize this subthreshold regime, which is so useful in many low-power IC designs, as simply being "off."

A root locus analysis [57] shows that for $k = 0$, the four poles lie at the same place on the real axis; as k is increased, the poles run along diagonal paths, with the two rightmost poles hitting the imaginary axis and theoretically inducing self-oscillation at $k = 4$. The ability of the filter to self-oscillate in a reasonably stable fashion can be creatively useful; various nonlinear and nonideal effects may cause practice to diverge from theory in interesting ways.

In our implementation, the amount of feedback is controlled by a second two-OTA amplifier hooked to the "bottom" of the ladder in Fig. 31. The first OTA converts the differential output voltage into an output current through its transconductance g_{m1}, and the second OTA converts that current into a single-ended voltage via its transconductance g_{m2}. The transconductance of an OTA operated in subthreshold is defined as [43]

$$g_m = \frac{\kappa I_{bias}}{2U_T}. \tag{34}$$

Writing Kirchhoff's Current Law at the node they share, we can derive the DC transfer function as

$$V_{out} = V_{cm} + \frac{I_{bias_1}}{I_{bias_2}} \left(V_{left} - V_{right} \right). \tag{35}$$

The effect of increasing the amount of negative feedback in our FPAA implementation is illustrated in Fig. 32. Note the peaking resonance behavior, as well as the changing DC gain.

Fig. 33 shows example time-domain system outputs for three cases of square wave inputs: $k = 1$, 10 kHz input (top graph), $k = 4$, 10 kHz input (middle graph), and $k = 4$, 100 Hz (bottom graph). Each waveform on a graph corresponds to a different value of V_{cutoff}. The DC components of the waveforms are shifted to provide a convenient way to show the outputs for several different cutoff frequencies on the same graph. Varying resonance effects can be seen in the Gibbs-like oscillations at input transitions.

4.5.1 How Many VCFs Would Fit?

The question of how many such VCFs could fit on any given FPAA platform is complicated. Of course, it would depend on what other kinds of components one might want to put on

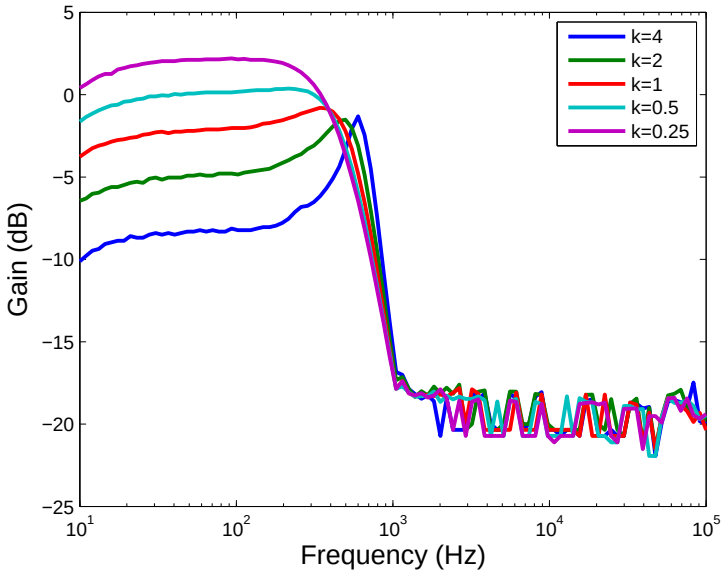


Figure 32: Measured VCF magnitude frequency response for different amounts of negative feedback k.

any given FPAA, but it would also depend on how much effort could be put into making clever use of FPAA resources and developing efficient routing schemes. RASP 2.9a could support 12 pFET-based ladder VCFs, assuming perfect utilization of all the available CAB pFETs in the chip; experience suggests that between 8 and 10 may be more realistic due to imperfect utilization. However, the chip also contains numerous nFETs, so one could invert the topology (making it look more like a traditional NPN-based ladder filter), yielding another 8 to 10 filters.

The RASP 2.9a CABs also contain MITEs (Multiple-Input Translinear Element Networks), which are multiple-input floating-gate pFETs; these could be potentially used to facilitate additional VCFs. Even more VCFs might be implemented if the floating-gate routing switches could be pressed into "double duty" as legs of the ladder, although that would likely be the most challenging approach.

In any case, even relatively straightforward implementations could readily support the VCF needs of a polyphonic synthesizer with a single FPAA chip.

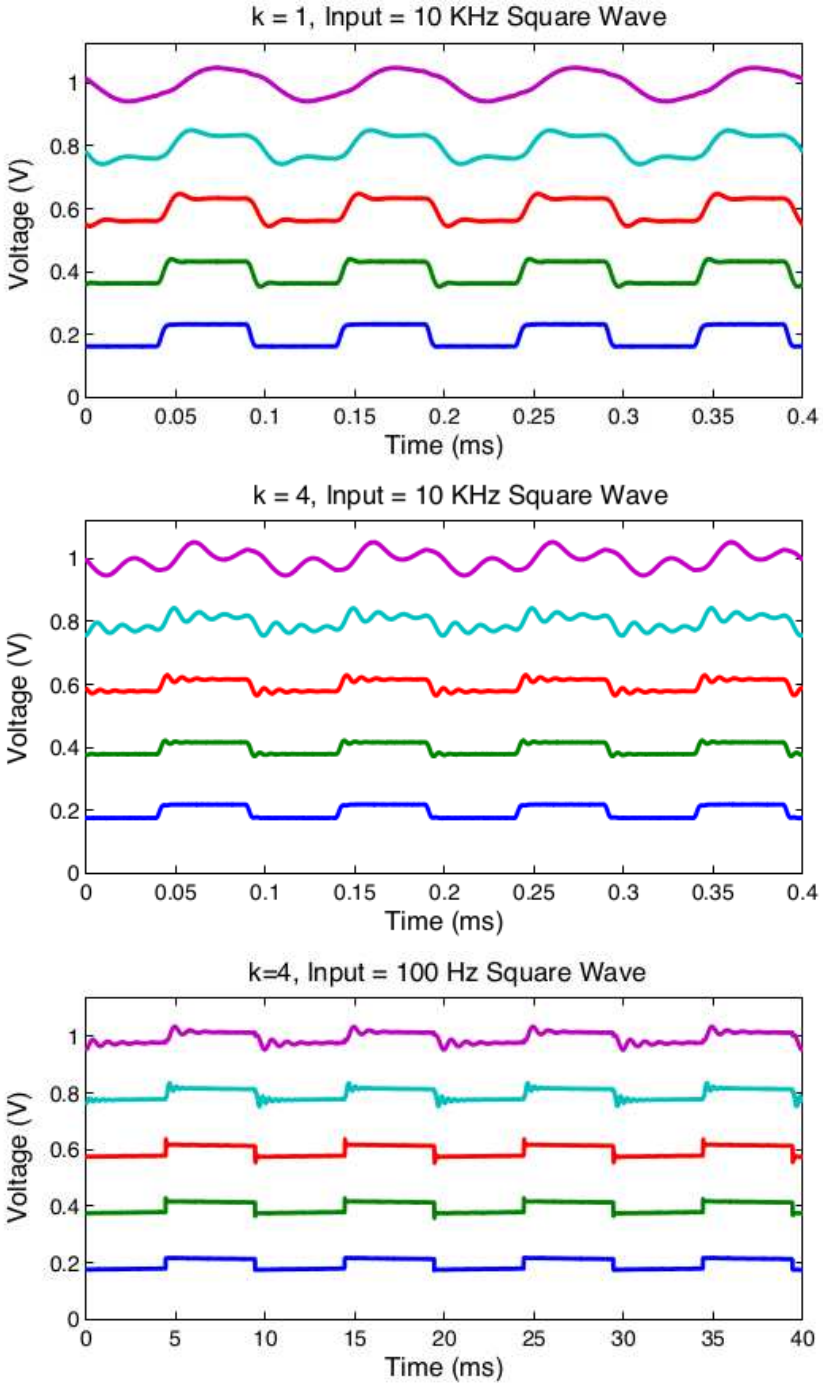


Figure 33: Examples of time-domain outputs; different in each graph correspond to different cutoff frequencies associated with different ladder bias currents, as set by V_{cutoff}. The different graphs correspond to different feedback amounts k and different input frequencies as indicated in the graph titles.

4.5.2 Differential amplifier implementation issues

The gain of the two 2-OTA[4] amplifiers used in the feedback path and the final output of the filter is controlled by floating-gate transistors, whose value must be programmed; this is inconvenient if smooth real-time changes are desired. The "built in" OTAs on RASP 2.9a have their gain controls "hardwired" to these floating gates. It would be more convent to directly control both the feedback amount and the differential amplifier driving the output; the latter then plays the subtractive synthesis role of a VCA. One could construct OTAs using the stand-alone generic pFETs and nFETs that are available in each CAB (three of one type of FET and two of another type could be made equivalent to one of the dedicated OTAs). Alternatively, one could use the Gilbert multipliers that are available in each cab; these may seem like overkill, since they are four-quadrant multipliers and only two-quadrant multiplication is needed, but they may be an attractive option if they are not being used for other options. We leave these issues as directions for future work.

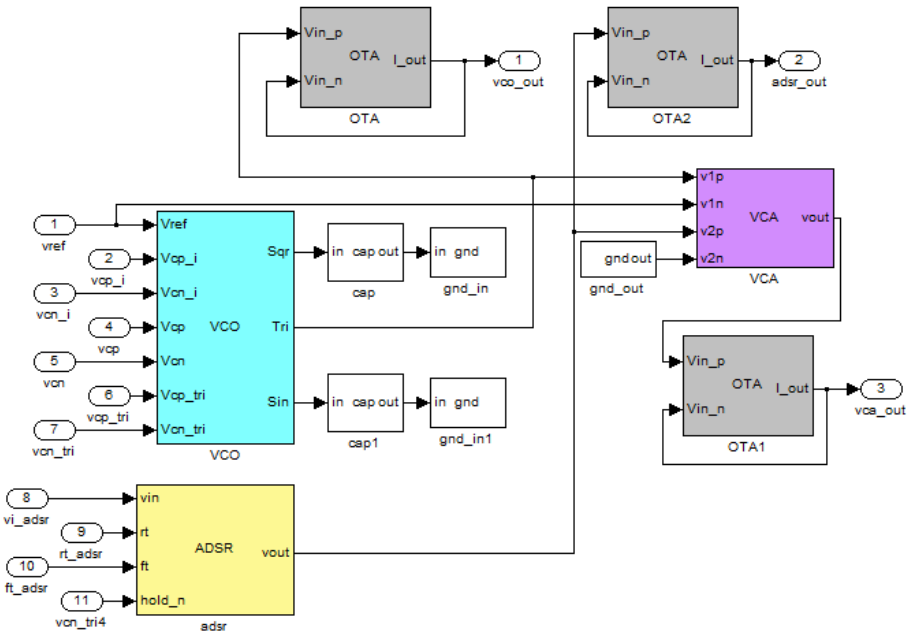
4.6 System Synthesis Results

Now that we have developed the fundamental blocks for our synthesizer, we will slowly build up a full-fledged synth using all of these components.

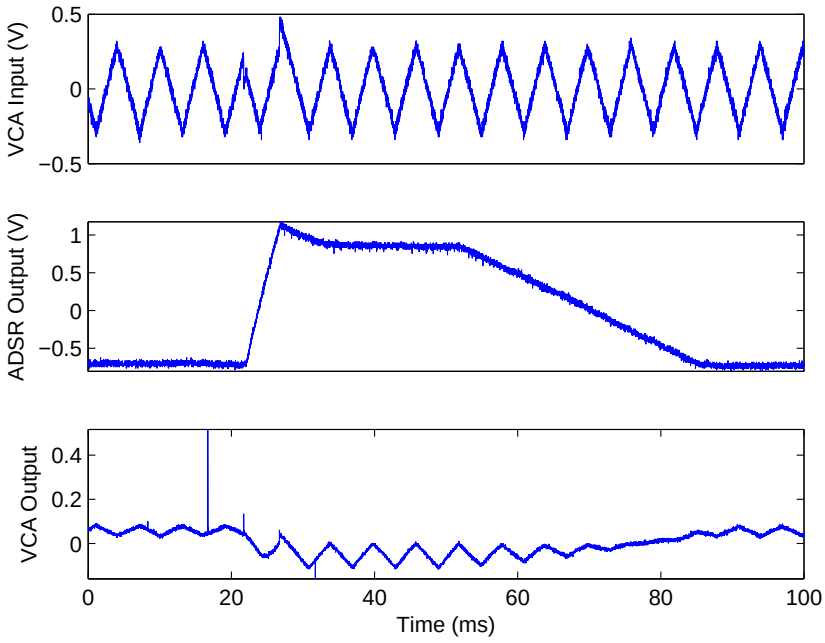
4.6.1 ADSR Envelope and VCA

First we demonstrate that an ADSR envelope can be applied to our VCA to create a plucked string sound. Fig. 34(a) shows the Simulink block diagram used for this experiment. The VCO creates a triangle waveform (top trace of Fig. 34(b)) which is fed into the VCA. The ADSR circuitry generates an envelope, shown in the middle trace of Fig. 34(b). These two signals are multiplied together in the VCA, resulting in the bottom trace of Fig. 34(b). The basic plucked string sound has been achieved. We see that, before onset of ADSR, the output of the VCA is suppressed. As soon as ADSR onset begins, the amplitude spikes up. Then it slowly decays down, similary to what one would expect with a plucked-string

[4]The construction "two 2-OTA amplifiers" looks like an error but is correct. Each amplifier uses two OTAs; the first turns voltage into a current, and the second essentially acts like a resistor turning current back into a voltage. The total gain of the two-OTA structure is determined by the ratio of the gains of the individual OTAs; having both gains available allows designers to make complex performance tradeoffs.



(a)



(b)

Figure 34: Demonstration that an ADSR envelope can be applied to a triangle waveform using our VCA. (a) Schematic diagram of ADSR and VCA combination. (b) Traces at various points in the system. The top trace is the triangle waveform generated by the VCO. The middle trace is the envelope waveform generated by the ADSR circuit. The bottom waveform is the resulting output of the VCA when the two signals are multiplied.

instrument.

We do see a nonideality in this result, however. It seems that the offset removal did not characterize out all of the offset, so that when the gain is set to be zero, we still see some of the input signal. To improve this result, we need to study which aspect of the offset removal is limiting the results. We may not have enough resolution in our measurements of the offset, our programming accuracy may not be high enough, or we may need to reconsider which devices should be characterized.

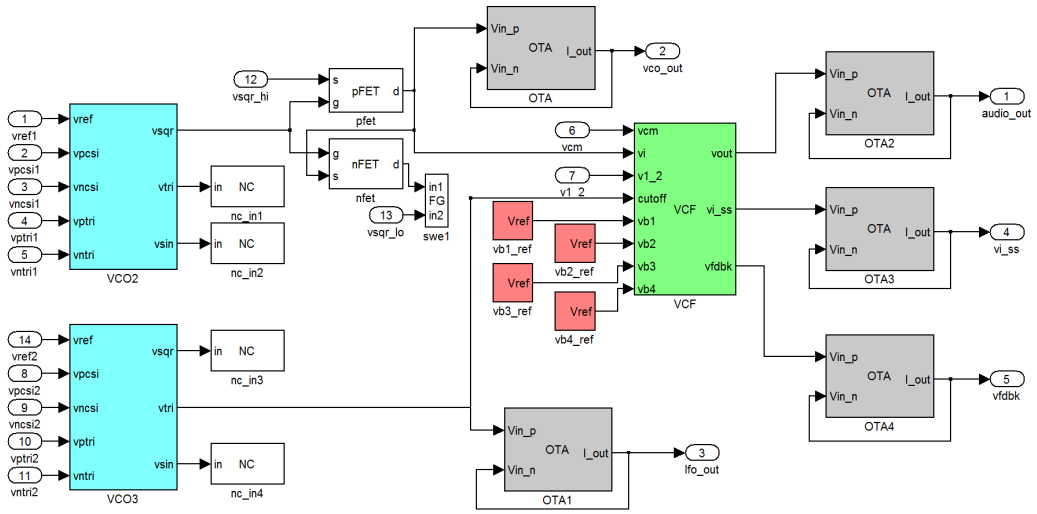
4.6.2 Demonstrating LFO

The main purpose of the Low Frequency Oscillator (LFO) is to modulate the cutoff frequency of the VCF. This enriches the sound with a "wah" effect. To implement this on the FPAA, we set up the experiment as shown in Fig. 35(a). The top VCO generates a fast square wave. We set the top and bottom rails of this square wave by using a second inverter whose rails are fed by DACs. This was done to account for the linear range of the filter circuit. We also wanted to be able to set the DC point of the square wave arbitrarily. A slow triangle wave is generated by a second VCO and fed into the cutoff voltage input of the top VCO.

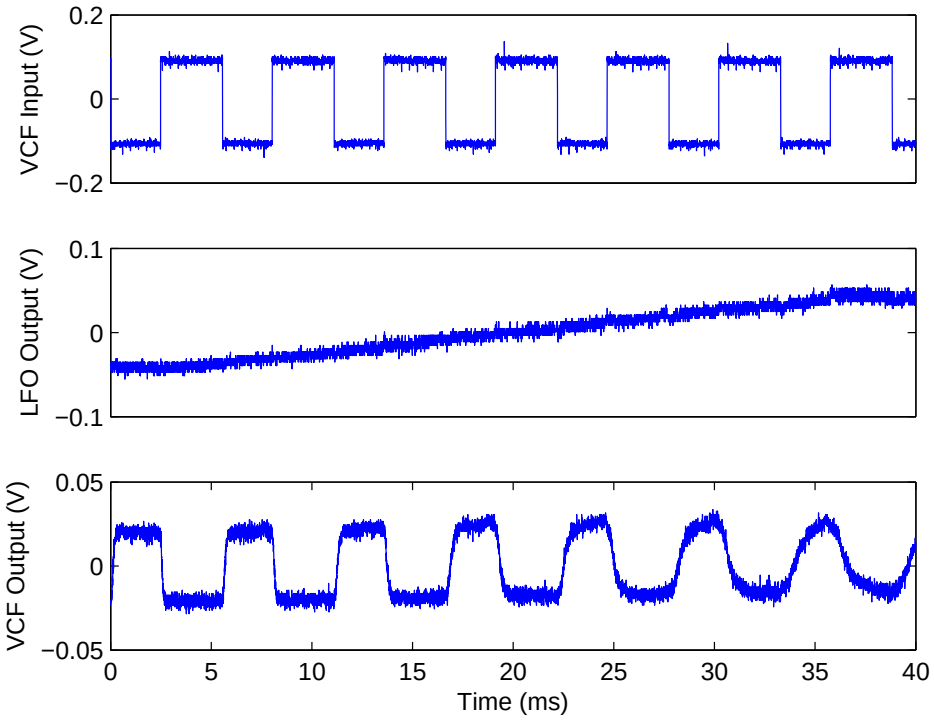
The resulting output is a square wave with varying amounts of its harmonics cut off, exactly as desired. As the triangle VCO continues to oscillate over many seconds, this will have the effect of a slowly varying "wah" sound, making the resulting waveform sound quite interesting.

4.6.3 A Subtractive Synthesizer

The final experiment is to connect all components together and create a synthesizer with a VCO, VCA, VCF, LFO, and ADSR. The schematic is shown in Fig. 36(a), and the results are shown in Fig. 36(b). First, we generate a square wave with the top VCO (not shown in Fig. 36(b)). Next, we apply an ADSR waveform to the square wave, as shown in the top trace of the figure. This results in an output which has no gain when ADSR is low, but has a significant amplitude during the onset of the ADSR. So we have a plucked-string sound. Next, we generate an LFO. This is shown in the third trace of the figure. The LFO and

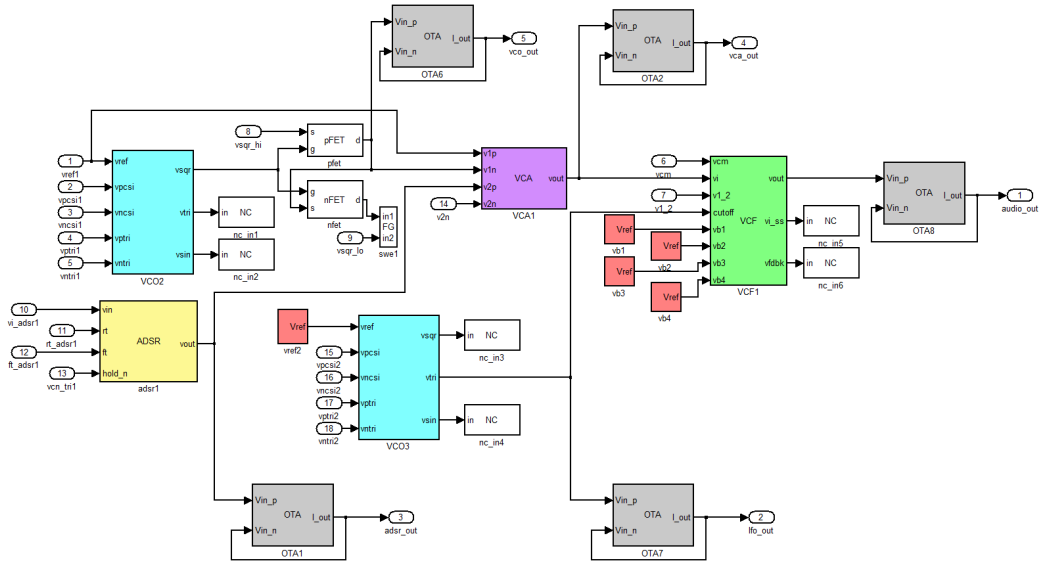


(a)

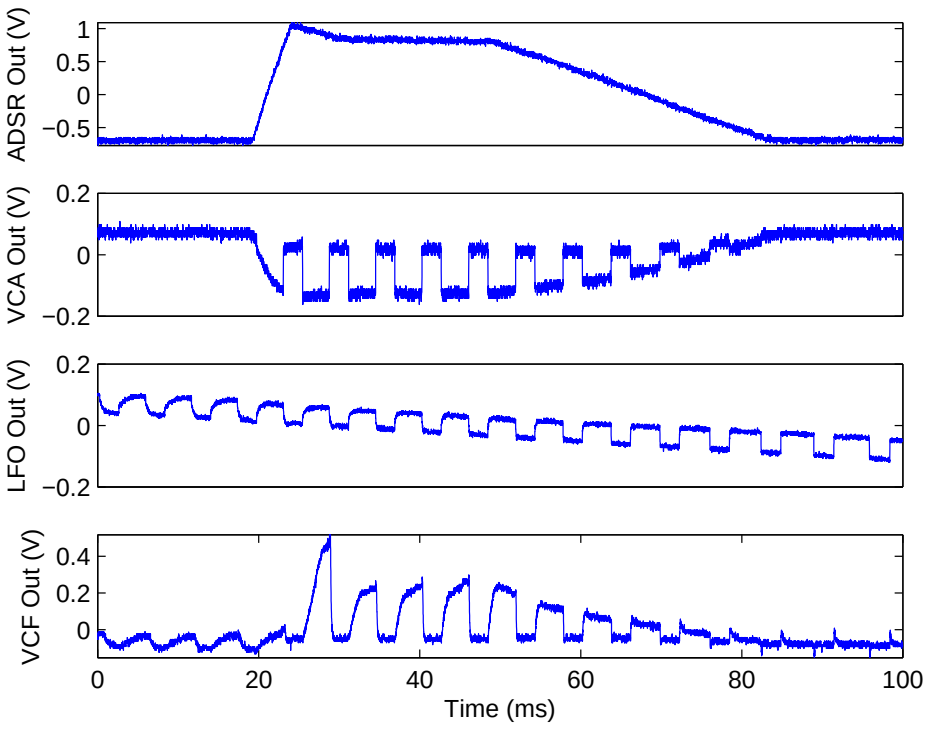


(b)

Figure 35: Demonstration that an LFO can be applied to a waveform using our VCF and a slow VCO. (a) Block digaram of LFO experiment. (b) Traces at various points in the system. The top trace is the fast square wave generated by the first VCO, which serves as input to the filter. The middle trace is the slower triangle wave which slowly varies the cutoff frequency of the filter. The bottom trace is the output of the filter.



(a)



(b)

Figure 36: A subtractive synthesizer consists of two VCOs, a VCA, and a VCF. (a) Simulink diagram of subtractive synthesizer. (b) Initial results from synth.

VCA output are then fed into a VCF, and the result is shown in the bottom trace.

We see two main nonidealities is this plot. First, the output of the VCA changes its DC voltage with the onset of ADSR. This is again due to the offset not being completely removed from the VCA circuit. Secondly, we see that there is significant coupling from the square wave VCO into the LFO VCO. To improve this, the two blocks can be placed far apart. The results have a number of interesting aspects. When the ADSR is off at time t=0, some signal is still getting through to the output because of the VCA offset. This signal looks more like a sinusoid than a square wave, indicating that it is being cut off by the filter. When ADSR is activated, we see the output jump as expected. Its shape is not obviously an RC shape, partially due to the coupling into the LFO from the oscillator. As time progresses, we do see the output look increasingly square, as we expect. Another interesting aspect is that the LFO waveform seems to be filtered as well, for it looks like an RC shape at t=0 and appears more square at t=100.

4.7 Conclusions and Future Work

The FPAA is a flexible platform that allows exploration of analog music synthesis. Its components are capable of generating all of the building blocks of subtractive synthesis – oscillators, amplifiers, and filters. New FPAA tools were developed as a part of this project, and they enable musicians to use Matlab's Simulink as a tool for both designing and characterizing their musical circuits.

We have demonstrated a proof of concept for analog synthesis on an FPAA. The developed blocks and corresponding results have the potential to open up a whole range of interesting systems and circuit techniques for investigation. Examples of potential topics are simpler control of waveform shapes and improving automatic calibration, discussions which are beyond the scope of this chapter. In a similar way, we see the initial Simulink blocks described in this chapter continuing to develop such that they are more usable for non-experts in circuit design. These approaches may empower an analog music synthesis renaissance due to the advent of large-scale, reconfigurable analog chips.

CHAPTER V

CALIBRATION AND TEST OF A NEUROMORPHIC PLATFORM

This chapter discusses the Neuron1D, a large-scale neuromorphic chip designed to simulate neural networks using analog circuits. We first introduce the architecture of the Neuron1D and discuss its various features. Next we describe the removal of offsets in a subsystem of the Neuron1D. Finally, we discuss some learning experiments run on the Neuron1D using its STDP synapses.

5.1 Introduction to the RASP Neuron 1D

The goal of neuromorphic engineering is to create hardware that can perform tasks similar to those of the brain in a highly power-efficient manner. The RASP Neuron 1D is a reconfigurable neuromorphic chip which builds upon years of neuromorphic research in the CADSP lab. Fig. 37 illustrates the basic goal of the chip. The top of the figure shows a simple diagram of a neuron – it consists of input synapses, a conductive medium that connects those synapses to the cell body, and ionic channels in the cell body. The synapses inject current into the neuron, and if they inject enough current, the ionic channels will respond with a stereotyped voltage waveform known as an action potential. This action potential is then transmitted to the output of the neuron, called the axon, where it is relayed to other neurons' synapses. Our brains consist of 100 billion of these units, connected together in dense networks of hundreds or thousands.

The middle of Fig. 37 shows how this biological structure is mapped to CMOS technology in our chip. The synapses are implemented by single pFET transistors with triangular voltage waveforms applied to their gates. As indicated below the schematic, these devices can change their weight according to the timing of input and output spikes to the neuron. The pFET synapses' currents are then summed and sent to a neuron circuit, implemented with two channel transistors and two filters, an Na bandpass filter and a K lowpass filter. The step responses of these filters are indicated below the neuron schematic. Our chip is

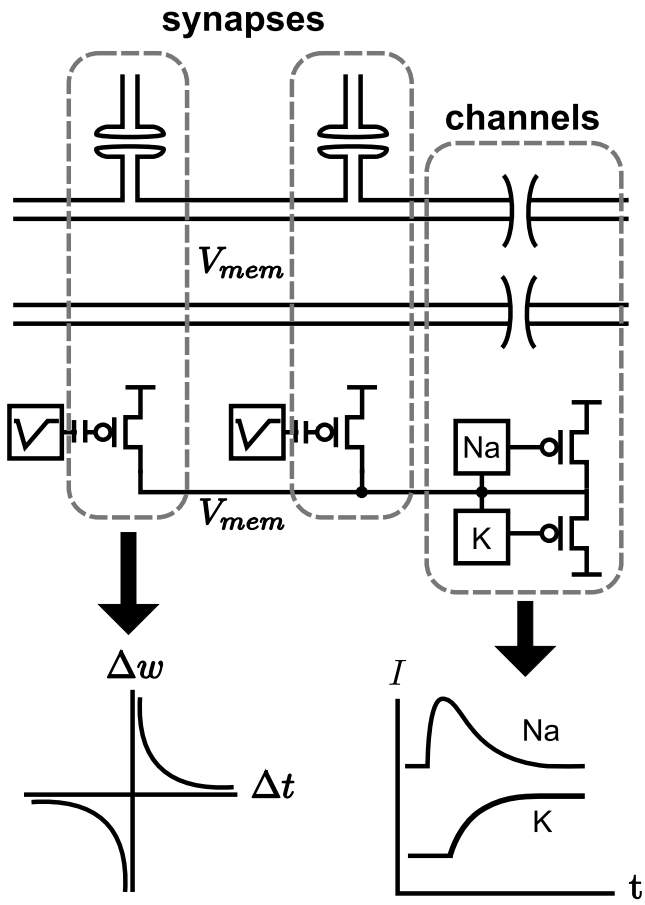
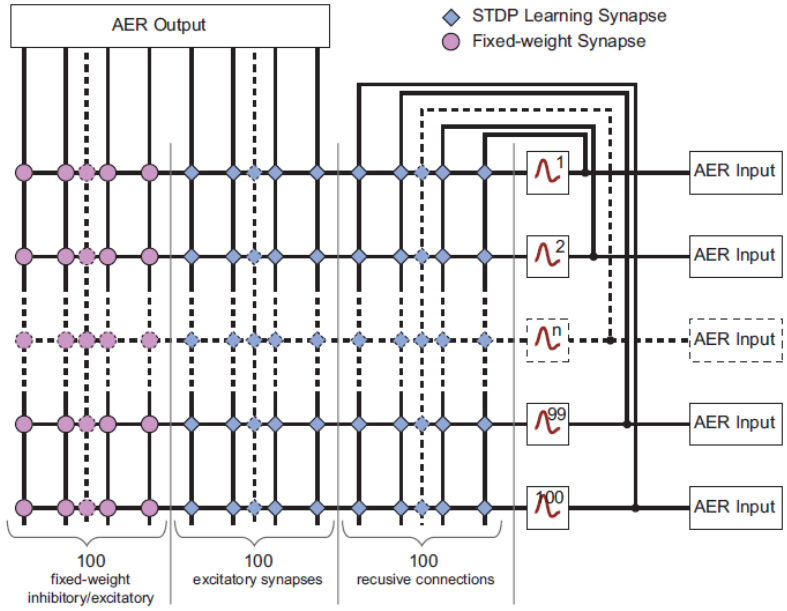


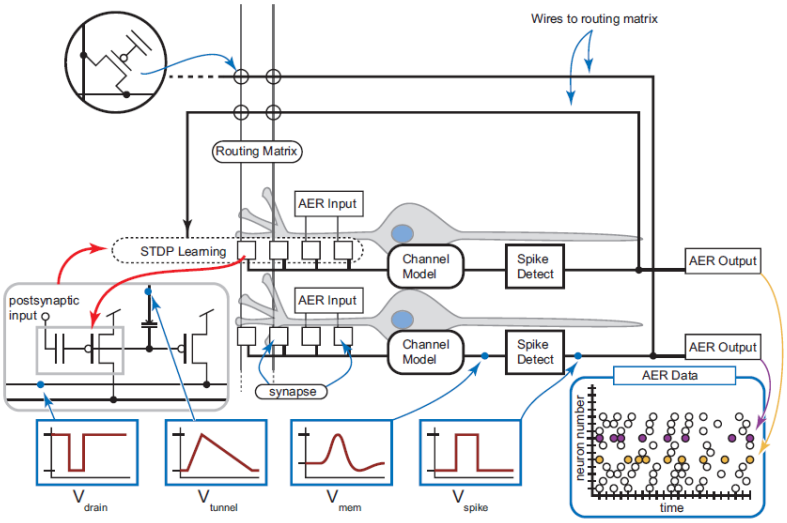
Figure 37: We have emulated a biological neural network (top) with a reconfigurable mixed-signal CMOS system (middle). We have taken inspiration from biology in two important ways. First, the floating-gate synapses of this system have an STDP learning rule, similar to that of biological synapses (bottom left). Second, the neurons in this system are modeled by pFETs gated by amplifiers which mimic the response of biological channels to a voltage clamp (bottom right). We will take advantage of these similarities to develop systems that learn in a manner similar to biological neural networks.

designed to implement the functionality of biological neural networks using these circuit components.

Fig. 38(a) shows the basic architecture of the chip [11]. There are 100 rows of neuron elements, each with 300 synapses (for a total of 30,000 synapses). 200 of those synapses receive inputs from an Address Event Representation (AER) circuit, which is a digital circuit that allows information about action potential "events" to be sent to this chip from another AER-enabled chip or MATLAB. Of these AER synapses, 100 can be inhibitory/excitatory but have a static weight after programming. 100 others are excitatory, and their synapse



(a)



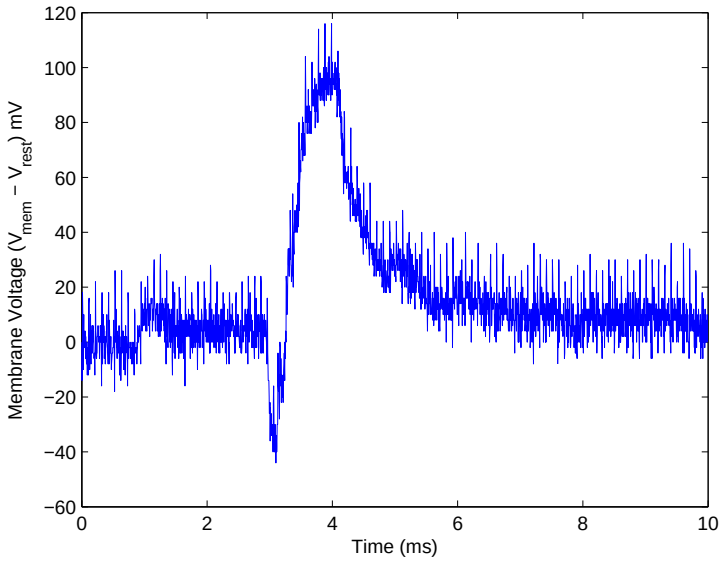
(b)

Figure 38: (a) High-level overview of Neuron 1D chip. (b) Detailed architecture of biological signals and channels in the Neuron 1D chip; both images from [13]

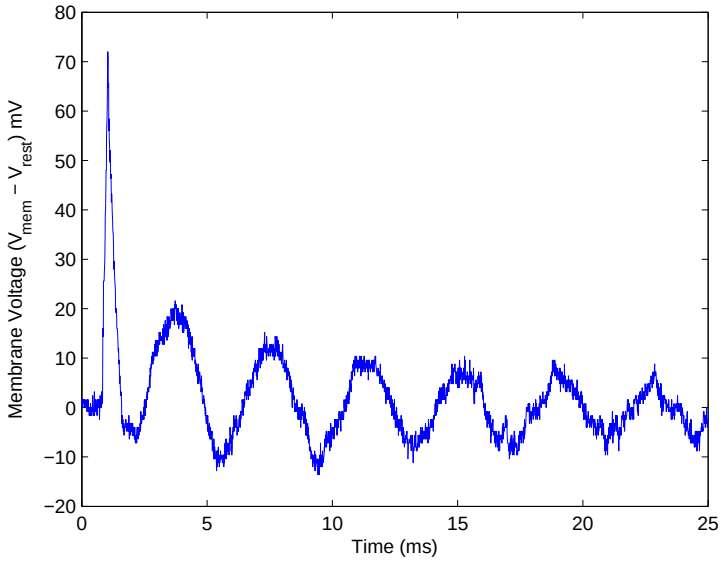
strengths can change based on a learning rule known as Spike-Time Dependent Plasticity (STDP). Finally, each neuron channel recurrently connects to all of the other neurons, so 100 more excitatory STDP synapses are available.

Fig. 38(b) shows details of how this architecture is implemented. Each synapse is a floating-gate transistor. When a signal is to be transmitted through the synapse, it receives

a triangle input waveform at its gate, which causes its output current to look like an Excitatory Postsynaptic Current (EPSC). The charge on the floating-gate determines how strong this current pulse will be. This concept is discussed in [20].



(a)



(b)

Figure 39: (a) Typical measurement from excited neuron. (b) Neuron exhibiting subthreshold oscillations.

A key component of the nervous system's flexibility is that it can adapt to changes in

its environment. Spike-Timing Dependent Plasticity (STDP) has emerged as one of the most promising biophysical explanations for learning at the synapse level [8]. This rule states that when a presynaptic neuron fires before a postsynaptic neuron, the synapse's strength increases by an amount which depends on the timing difference between the firings. Conversely, if the postsynaptic neuron fires first, the synapse's strength decreases as a function of the time difference. STDP synapses have an extra component which allows them to learn. These synapses are an evolution from the original learning transistor, introduced in [23]. In our STDP synapses, floating-gate charge is modified by a second, indirect transistor. According to the learning rule, the drain and tunneling voltage of this indirect transistor are pulled low and high, respectively, causing varying amounts of hot-electron injection and Fowler-Nordheim tunneling to modify the floating-gate charge. Thus the weight of the synapse changes. This is discussed in [48].

The core processing elements of the chip are 100 bio-physically inspired silicon neurons [17]. These differ from the original circuits because they contain floating-gate transistors to allow trimming of various parameters of the neurons. We typically operate these neurons in an "excitable regime," where a sufficiently strong pulse of current causes the neuron to generate a spike in response. An example of a typical measured response from one of these neurons is given in Fig. 39(a). We have noticed some interesting dynamic properties of these neurons. For example, one measurement showed a neuron undergoing subthreshold oscillations (see Fig. 39(b)), which is generally seen in neurons near a bifurcation.

5.2 Offset Removal in the Neuron 1D Gate Waveform Shaping Circuitry

The EPSC generated by a synapse depends on its floating-gate voltage and the waveform that is applied to its gate. This waveform is generated by gate waveform shaping circuitry, which is shown in Fig. 40. When the membrane potential V_{mem} becomes greater than some user-determined threshold V_{thresh}, a comparator trips. The comparator output goes into an edge detect circuit, which generates a pulse whose length is determined by half of a current-starved inverter. This pulse then goes into a full current-starved inverter. The biases for all three starved current sources are generated by floating-gate pFETs. So the

risetime, falltime, and pulsewidth of the system components are all determined by these three floating gates.

While biology demonstrates a wide degree of variation in its synapses, we would like to set up our system such that this variability can be added if desired. We may want variability with a certain statistical distribution, so having control over these parameters is crucial. The circuits in the gate waveform shaping circuitry are perfect for offset removal because they have a simple operation. Both the edge detection and the current-starved inverters are based on the slewing operation of a capacitor with a fixed current being applied to it:

$$\frac{\partial V}{\partial t} = \frac{I}{C}. \tag{36}$$

We know that this fixed current has an exponential relationship to the floating-gate voltage of the pFET

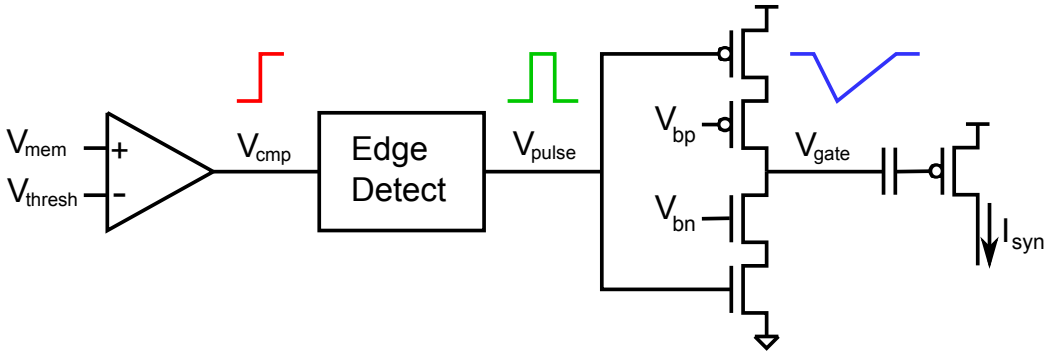
$$I = I_0 e^{\frac{V_s - \kappa V_{fg}}{U_T}}. \tag{37}$$

The falltime, risetime, and pulsewidth of the circuitry are all directly proportional to the slew rate of these capacitors. The slew rates are exponentially dependent on the floating-gate potential. Therefore we can assume that all of these parameters follow the relationship given below:

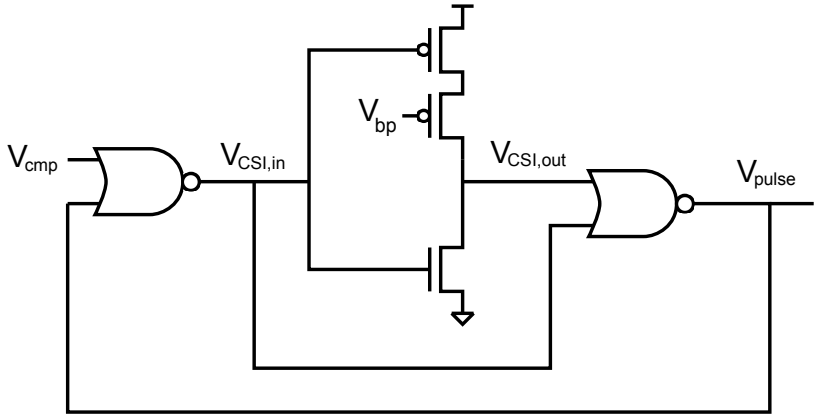
$$param \propto I_{0_{param}} e^{\frac{\kappa_{eff_{param}} V_{fg}}{U_T}}, \tag{38}$$

where $I_{0_{param}}$ is the scaling factor for each of the three parameters (one of $I_{0_{risetime}}$, $I_{0_{falltime}}$, or $I_{0_{pulsewidth}}$). The V_s term dropped out because it is assumed to be equal to V_{DD} and is incorporated in the $I_{0_{param}}$ term. We verified this exponential dependence for all three parameters, and this is shown in Fig. 41.

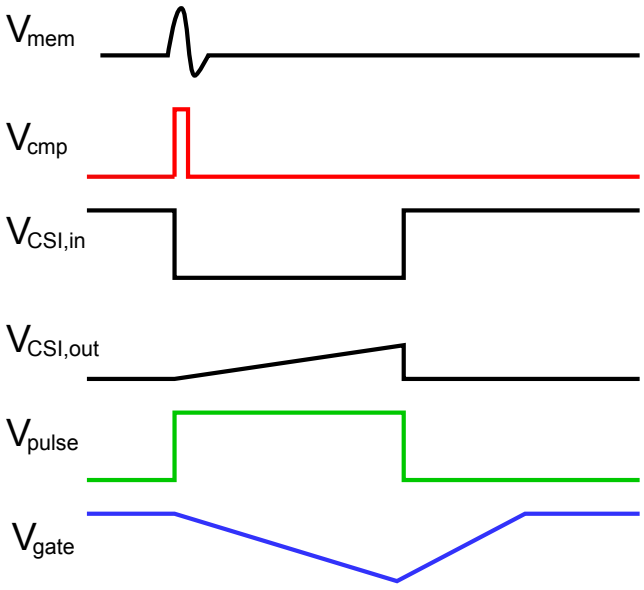
We assumed that κ does not change much from pFET to pFET. This means that we can extract κ_{eff} once, and then find $I_{0_{param}}$ for each channel. To find κ_{eff}, we program one channel to different floating-gate voltages and measure the slope of the resulting response. To find $I_{0_{param}}$ for a channel, we only program it to one floating-gate target, measure the



(a)



(b)



(c)

Figure 40: (a) Gate waveform shaping circuitry. (b) Gate-level schematic of edge detect circuitry. (c) Timing diagram of gate waveform shaping.

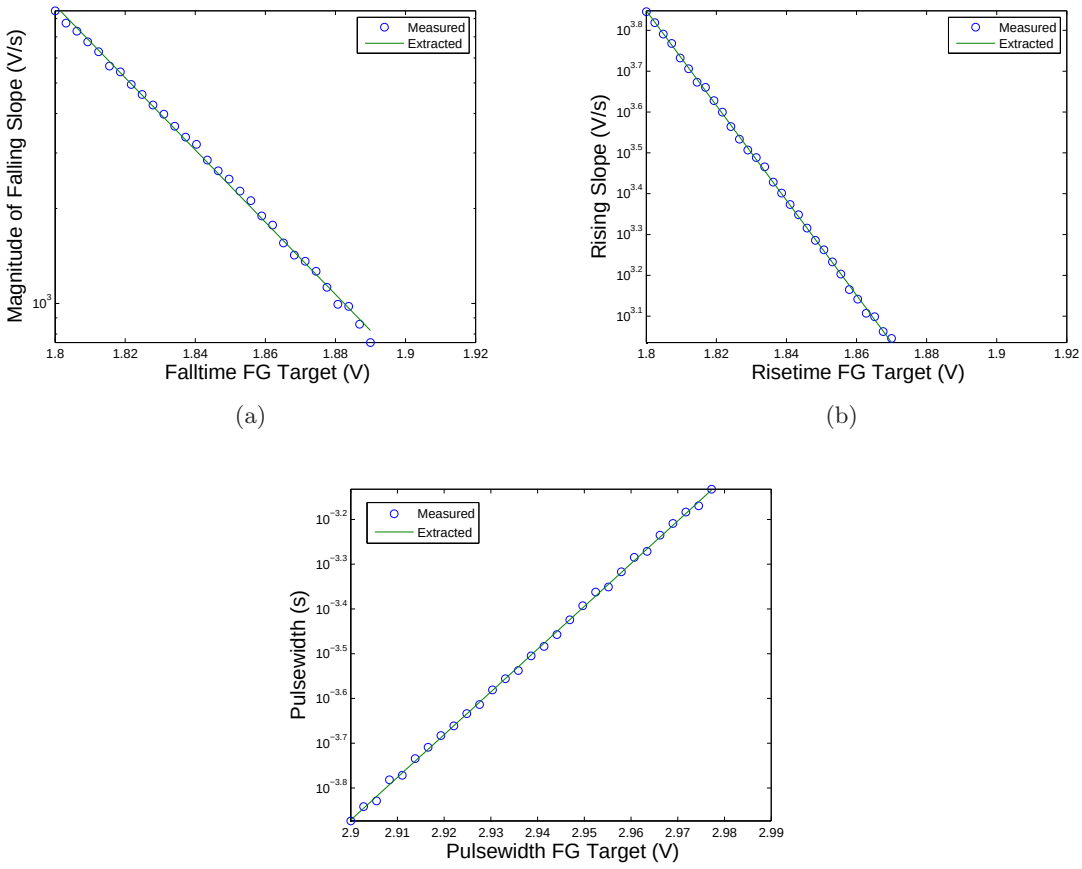
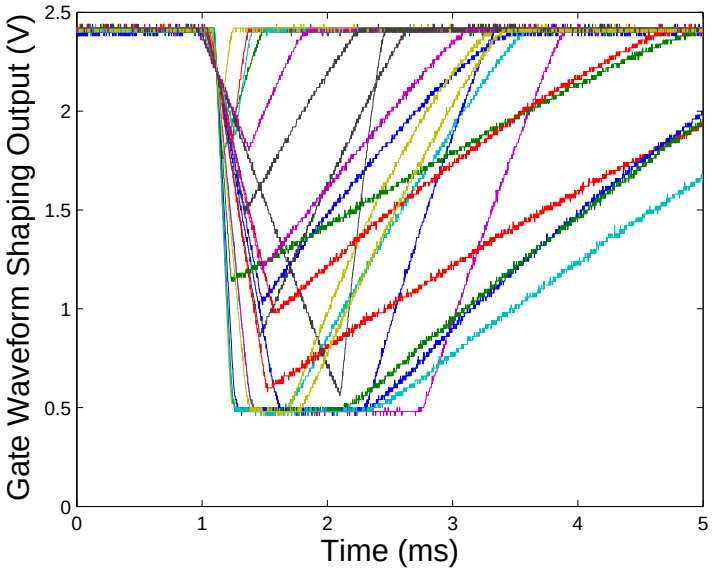


Figure 41: Dependence of (a) falltime, (b) risetime, and (c) pulsewidth on the floating-gate voltage in timing circuitry on the Neuron1D.

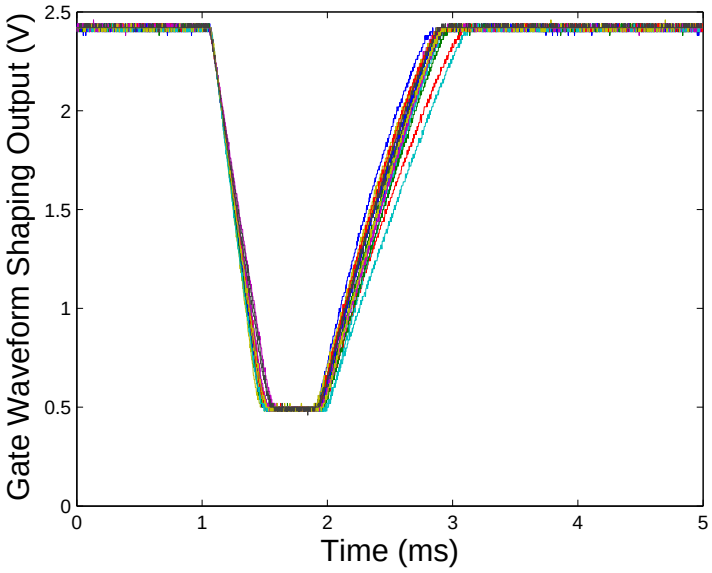
parameter value, and extract what $I_{0_{param}}$ must be from Eq. 38. Once we have κ_{eff} and $I_{0_{param}}$, we can achieve any waveform shaping parameter that we desire. The results of performing this experiment on a small subset of the gate waveform shaping circuits are shown in Fig. 42. These results are for the channels which were programmed correctly.

The statistics of the waveform characteristics have been drastically improved. The histograms of the four waveform characteristics are shown in Fig. 43.

After we performed initial experiments on the Neuron1D, we started to use the STDP circuitry to implement small networks. The following discusses those small networks, and was published in [46].



(a)



(b)

Figure 42: Output of waveform shaping circuits (a) before and (b) after offset removal.

5.3 Floating-Gate STDP Synapse

An synaptic STDP rule can be enforced by applying the correct waveshaping circuitry [47] to the terminals of a floating-gate transistor, shown schematically in 44(a). A simplified

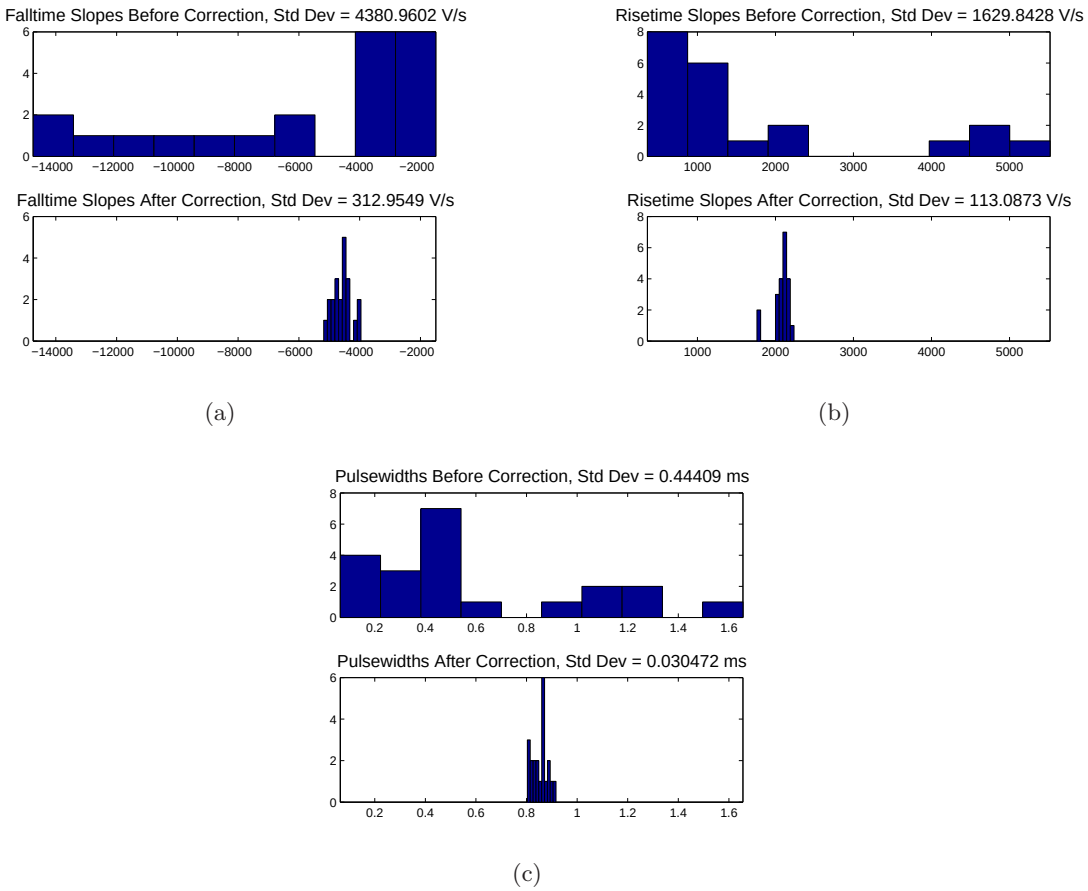


Figure 43: Histograms of (a) falltime, (b) risetime, and (c) pulsewidth before and after offset correction.

timing diagram is shown in Figure 44(b). These circuits cause potentiation when the presynaptic spike occurs before the postsynaptic spike and depression when the order is reversed. Potentiation is accomplished with hot-electron injection, which reduces the floating-gate voltage; depression is accomplished with Fowler-Nordheim tunneling, which increases the floating-gate voltage.

We achieve injection and tunneling with three waveshaping circuits. Whenever the presynaptic neuron fires, it pulses the synapse's gate low. A firing of the postsynaptic neuron forces a drain pulse followed by a delay and then a tunneling pulse. When a pre-post spike pairing occurs, the drain is pulsed low at the same time the gate is low, and therefore injection (and thus potentiation) occurs since the device conducts current at a high drain to source voltage. When a post-pre pairing occurs, the drain pulse does not have

any effect on the floating gate, and the tunneling pulse dominates the behavior, causing depression.

This synapse differs from that of [38] in that its tunneling and injection amplifiers are used for every pre and post synaptic spike, not just when they exceed a threshold. The waveshaping circuits are also tunable over a wide range of parameters, offering more flexibility.

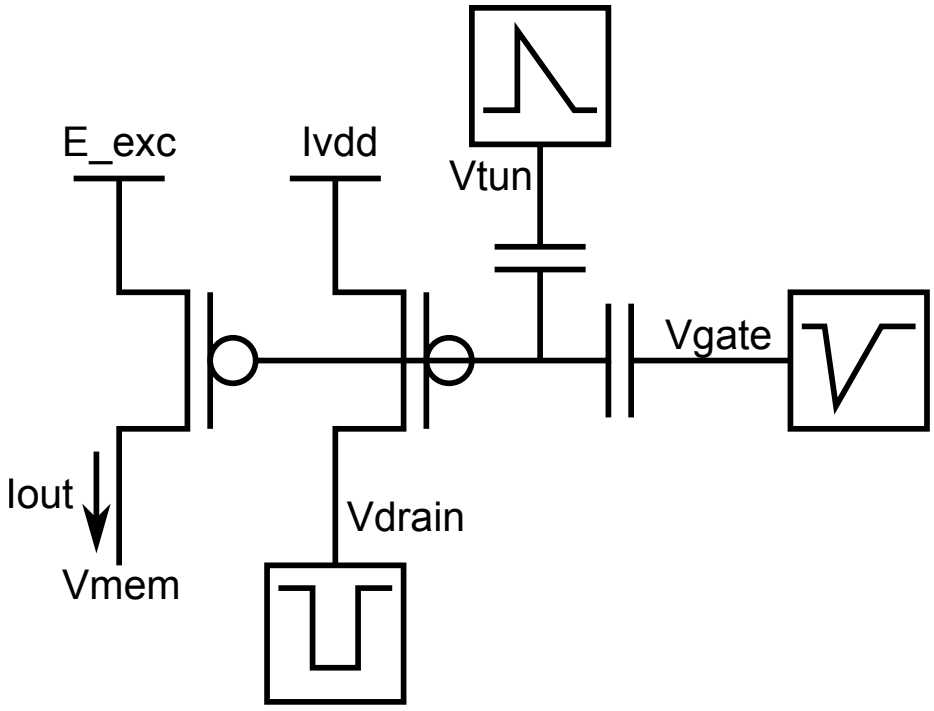
An STDP curve is composed of two parts. The portion showing how the neuron's strength increases with timing is called Long-Term Potentiation (LTP). The portion showing a weight decrease is called Long-Term Depression (LTD). We can control LTP and LTD using different tunneling and injection voltages, as well as the timing circuitry's parameters. For example, Figure 45(a) shows how varying the injection V_{dd} increases the rate of LTP. Figure 45(b) shows that we can achieve the classic STDP curve when LTP and LTD are combined.

Typical rise and fall times for the waveshaping circuits are on the order of a few milliseconds. In [11], for example, the drain pulse is 2 ms, gate pulse is 6 ms, and the tunnel pulse is 8 ms. A model of how these times affect the STDP window is derived in [47]. Due to the tunneling amplifier in this chip, our tunneling pulse is exponentially decaying rather than linear, which should be taken into account when evaluating the model.

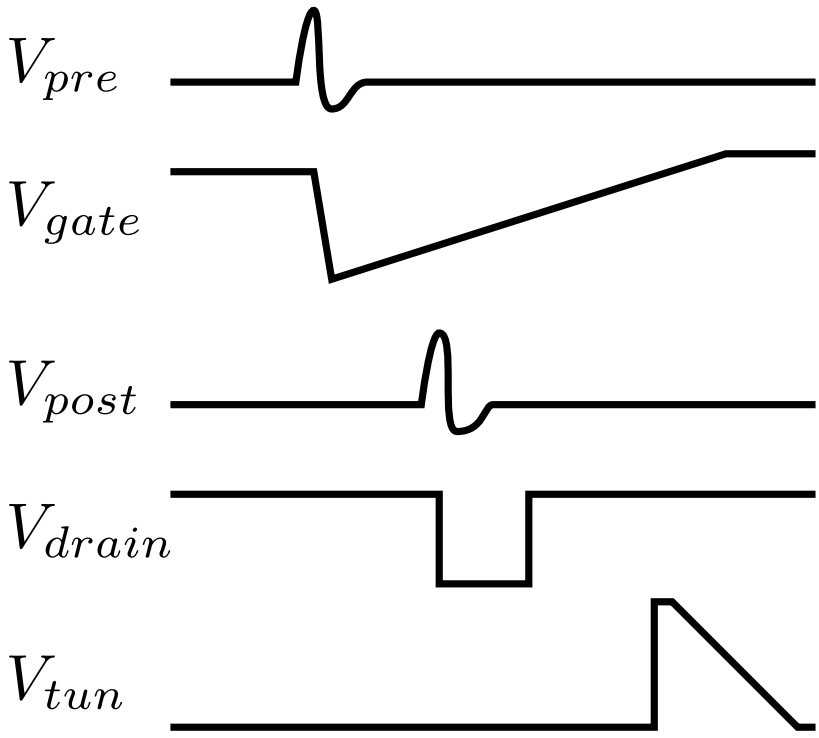
The analysis in [47] shows that the weight change is a function of the current weight. [30] notes that multiplicative STDP rules result in bimodal weight distributions (in a rate-based system) if both LTP and LTD depend on the weight, and depression dominates. We observed some degree of bimodality in our experiments – typically, one synapse would eventually dominate the others. [30] note that a computational result of bimodality is competition amongst synapses, and in the following experiments we exploit this competition to perform different computational tasks.

5.4 Small STDP Network Tests

Now that we have shown simple STDP curves for a single synapse, we employ these devices in small networks.

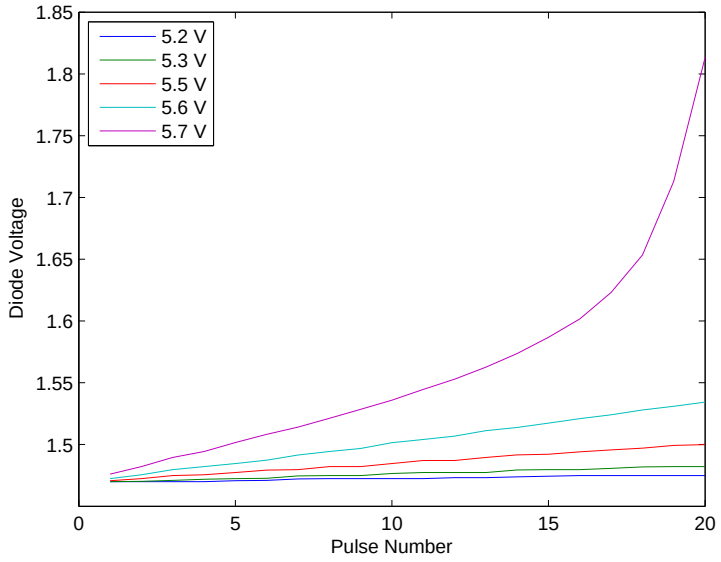


(a)

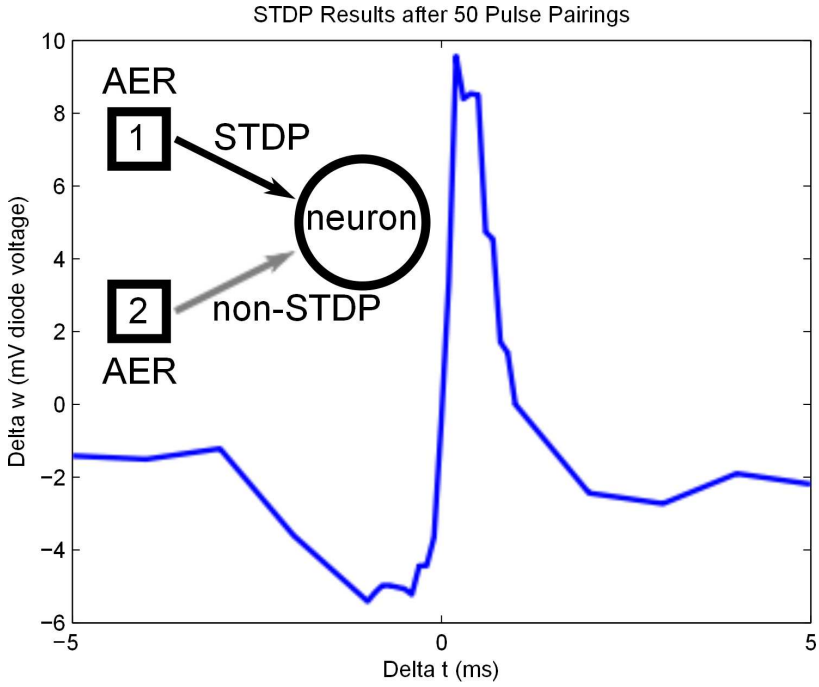


(b)

Figure 44: The synapses on this chip implement an STDP rule. (a) Schematic of waveshaping circuitry on the chip. (b) Timing diagram for the waveshaping circuitry. When the presynaptic spike fires, the synapse's gate is pulsed down in a triangle shape. When the postsynaptic spike fires, the indirect device's drain is pulsed down (causing injection), and some time later its tunnel line is pulsed up (causing tunneling).



(a)



(b)

Figure 45: Illustration of the basic functionality and control of the STDP synapse. Weight change is measured by a diode I/V converter. (a) The rate at which synapses are potentiated can be controlled by the injection voltage. For a particular gate waveform shape, a higher injection voltage will result in more change in weight. (b) Change in weight as a function of the pre-post synaptic firing times. We observe a positive change in weights when the post follows pre, and negative change when pre follows post. The large change at 5.7V is likely due to a change in the injection characteristic, since its shape changes with floating-gate voltage.

5.4.1 Weight Initialization

Initializing the synaptic weights in this system was accomplished heuristically. Since the synapses are indirectly-programmed floating gates, there will be some mismatch between the device that is programmed and the in-circuit device [56]. Additionally, there will be some mismatch in the gate waveform shaping circuitry. These two issues mean that the same floating-gate voltage on two different FG's may not result in exactly the same excitatory post-synaptic current (EPSC).

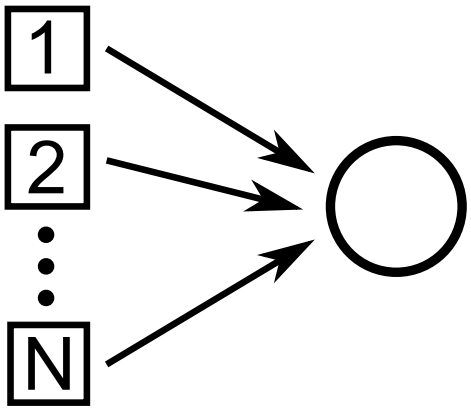
To overcome this problem, we determined the weights experimentally. We looped over each synapse and determined what weight would be necessary to cause the neuron to spike. Then we programmed the synapse to a few millivolts higher than that value, which sets the EPSC amplitude lower than the threshold for causing a spike. We refer all plots of weights in this paper to this initial value. We will track the change in floating-gate voltage rather than the absolute voltage to get a sense of how the EPSC is changing.

5.4.2 Strengthening Coincident Events

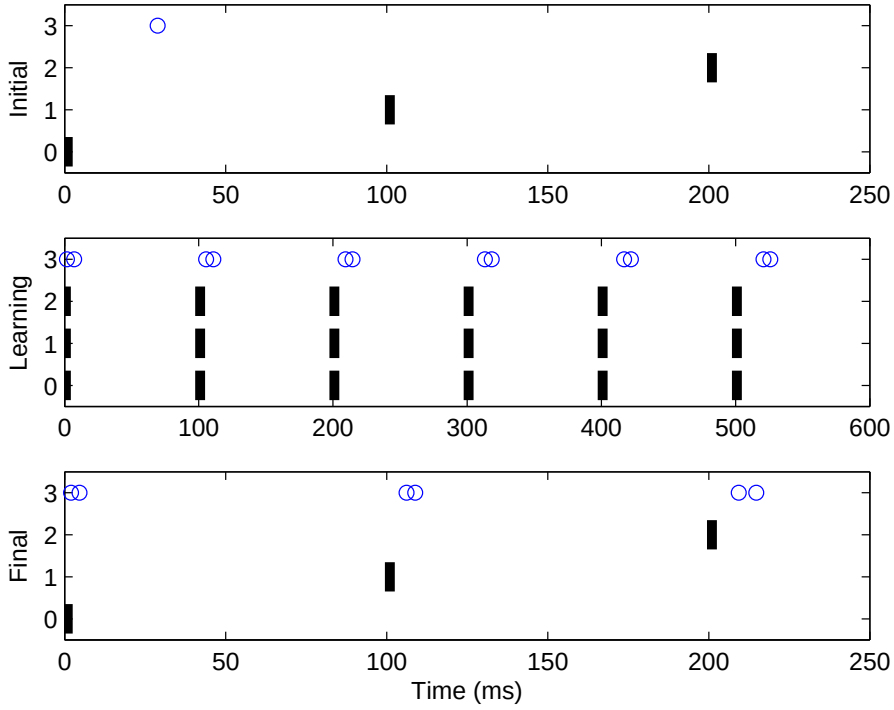
The experimental setup for our first experiment is shown in Fig. 46(a) with N = 3 inputs. We apply these inputs to our circuit through synapses connected to a communication interface known as Address-Event Representation (AER). We test LTP by strengthening all of the synapses. Initially, none of the inputs can cause the neuron to spike by itself. To get the neuron to learn, we fire all three AER inputs at the same time. Their EPSC's sum, which is enough to cause a spike. Since the timing between these inputs and the output spike is short, we expect all the synapses to be potentiated. After repeated coincident events, all of the synapses are potentiated enough that each one has a strong enough EPSC to cause the neuron to spike.

5.4.3 Poisson Input Trains with a Refractory Period

For the next few networks, we wanted to view the system response to Poisson-distributed input firing intervals. Using a pure Poisson distribution adds an interesting aspect to the behavior of the synapses. If an input event arrives at the triangle generator before the



(a)



(b)

Figure 46: A simple test of STDP consisted of a neuron and several AER inputs. (a) Schematic of experimental setup. (b) Result of learning. A black bar represents the time at which an input was applied to a neuron. A blue circle represents when an output spike was detected. Initially, none of the AER channels causes the neuron to spike. When they are all pulsed simultaneously, they cause a spike in the output, which potentiates all of them. Only a portion of those events are shown in this figure. Over time, this increases each synapse's strength such that it can cause the neuron to spike on its own.

triangle has finished rising to its pre-spike value, the resulting waveform will dip below the original minimum. This increases the amount of current during the drain pulse and the oxide voltage during tunneling. We found that this resulted in a rapid strengthening of the synapse, meaning that the injection process dominated. This process has is similar to the phenomenon of paired pulse facilitation that occurs in biology.

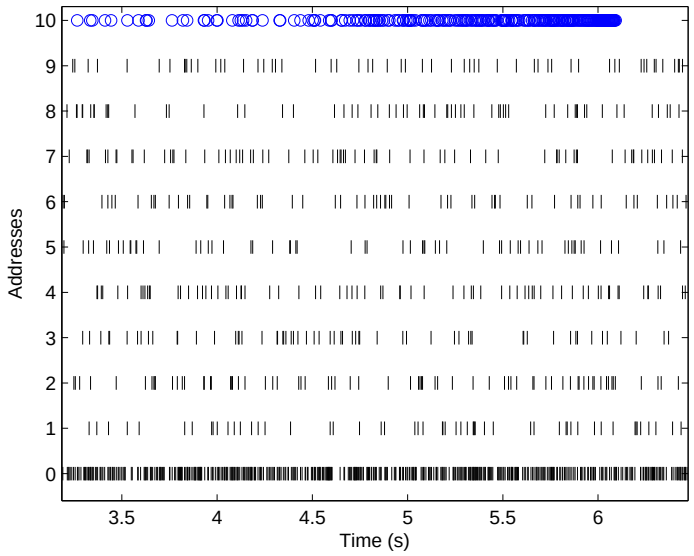
To prevent paired pulse facilitation from dominating the learning, we constrained the inputs to enforce a refractory period. After generating a Poisson input train, we post-processed the train and removed any spikes that occurred within a short time window of another spike for the same channel. This value ranged between 2 and 4.5 ms.

5.4.4 Learning the Highest-Frequency Input

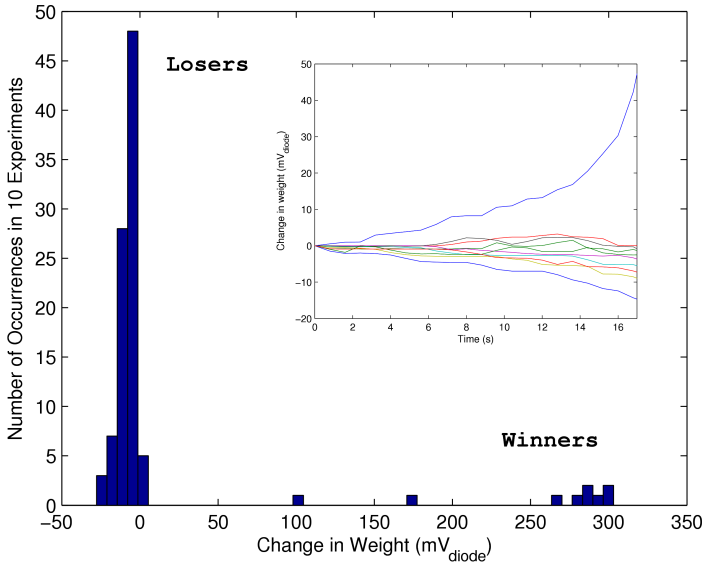
Our second test was to set up the network to learn which input was the strongest. We used the same network as the previous experiment, except that N=10. Nine inputs were given a modified Poisson input distribution of 20 Hz, and one was given a much higher rate of 200 Hz. All weights were initialized to a value just below their firing threshold. A coincidence of two inputs will result in an output spike, and both synapses will be potentiated. We expect that the input with the fastest rate will be a part of these potentiations more often than any other input, so it will dominate.

Fig. 47(a) shows a portion of the raster plot for this experiment. All inputs are firing at the same rate except for channel 0. Initially, the output neuron is firing at a rate which is lower than that of channel 0. Over time, the synapse strengthens, and eventually channel 0 causes the neuron to fire every time it supplies an input. Eventually, channel 0 becomes so potentiated that its floating-gate voltage forces the membrane high, regardless of whether the gate waveform has been activated. This means that no more events are detected, since the comparator is forced high.

This behavior is similar to the behavior of a Winner-Take-All (WTA) network [10], which has been identified as a computationally powerful tool [41] and has been used successfully for competitive behavior in VLSI systems [39]. This network demonstrates WTA-like behavior in terms of the weight modifications. This is shown in Fig. 47(b). We ran the test 10



(a)



(b)

Figure 47: An STDP network designed to strengthen the neuron with the highest firing rate. 10 STDP-enabled AER inputs are connected to the same neuron. One input's rate is 100 Hz, while all others are 20 Hz. (a) Typical raster plot. The output neuron's spike rate increases to that of the highest-frequency input until it is forced to be activated all the time. (b) We ran the experiment 10 times, designating a different "winning" channel each time. After tuning the gate waveforms and synapses correctly, the highest-frequency input usually wins. The histogram shows how often a change in weight appears in the result.

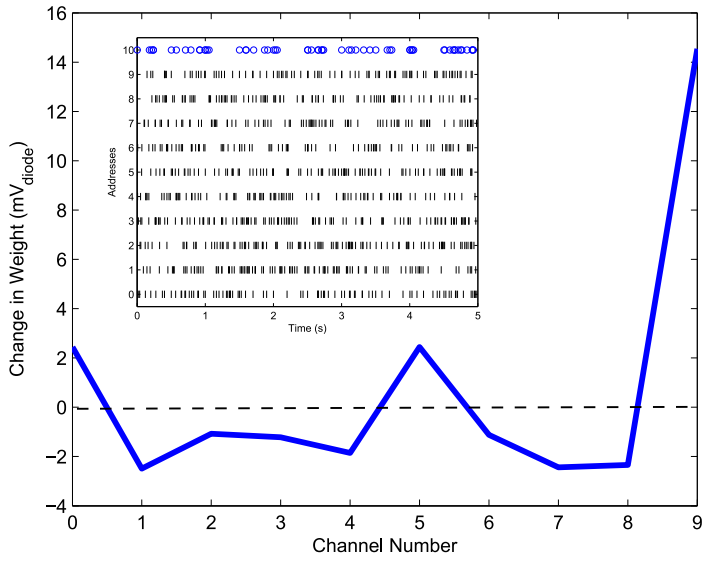


Figure 48: The network learns which inputs are synchronized, even if noise is present. Channels 0, 5, and 9 were synchronized at 5 Hz, and then a 15 Hz Poisson spike train was added onto each of the synchronized channels. All other channels were set up as 20 Hz Poisson spike trains. A refractory period of 2 ms is enforced. After a short period of learning, the weights which have changed the most are channels 0, 5, and 9.

times, and each time a different input channel had the highest firing rate. Given the correct learning parameters, we saw that the input with highest frequency usually showed the highest change in floating-gate voltage, and it saturated the neuron. In some cases, channels which did not have the highest rates would win. This was usually caused by a combination of LTP's runaway nature small mismatches from channel to channel.

5.4.5 Identifying Synchronized but Noisy Inputs

Our next task was to see whether the network can learn to prefer coincidences in the presence of uncorrelated inputs and noise. For our 10-input system, three inputs were synchronized at a low frequency, and high-frequency Poisson noise was superimposed over them and the other inputs. This is similar to the experiment performed in [42], where spatiotemporal spike patterns were found via STDP amongst distractor spike trains. Our neuron learned which inputs were synchronized, as shown in Fig. 48. This figure shows the weights early in the learning process. As the weights evolve further, one synapse will eventually dominate the others.

5.5 Conclusion

In this chapter we developed a method to remove offset in the waveform shaping circuitry of the Neuron1D. We also presented tests of the Neuron1D's STDP synapses. We showed that a neuron connected to a group of STDP-enabled inputs can learn which input is spiking fastest, as well as which inputs are synchronized, even in the presence of noise. We also discussed practical considerations in the creation of networks, such as weight initialization and refractory periods. In the future, we hope to build upon these simple networks to create more complex architectures which perform more complicated computations.

CHAPTER VI

DESIGN, LAYOUT, TEST, AND APPLICATION OF A CURRENT-MODE DAC FOR A DIGITALLY-ENHANCED FPAA

Proponents of analog signal processing state that analog systems should act as the frontend for many applications since they offer low-power solutions to simple computations. However, some problems are better-suited for digital systems, such as control, short-term memory, and state machines. So digital systems are often needed to accompany these analog frontends.

An important topic in reconfigurable systems research is how to communicate information between analog and digital subsystems. The RASP 2.9v is a new FPAA with enhanced capabilities for interfacing with digital systems. This work involved the design, layout, and testing of a portion of the RASP 2.9v: a Computational Analog Block whose purpose is to act as a reconfigurable current-mode Digital to Analog Converter (a "DAC CAB"). In the following sections we will briefly introduce the major features of the 2.9v chip, describe the design considerations for the DAC CAB, and then show experimental results from a current-mode DAC compiled on this chip.

6.1 Introduction to the RASP 2.9v

An overview of the intended use of the RASP 2.9v is shown in Fig. 49. The CADSP lab has developed software which allows users to create Simulink blocks which represent components of the CAB. These blocks are translated into SPICE netlists by a program called Sim2Spice, and these netlists are in turn translated into lists of floating-gate targets by GRASPER [33].

The hardware platform is a test board, which houses a microcontroller, DAC, ADC, and the FPAA. The floating-gate targets from GRASPER are programmed using an on-chip programmer, with programming commands sent from the on-board microcontroller. Analog inputs can enter the FPAA from DACs located on the test board or from the external environment. These analog signals are processed using CABs similar to those of the RASP 2.8a. Analog outputs are sent to an on-board DAC and relayed back to the

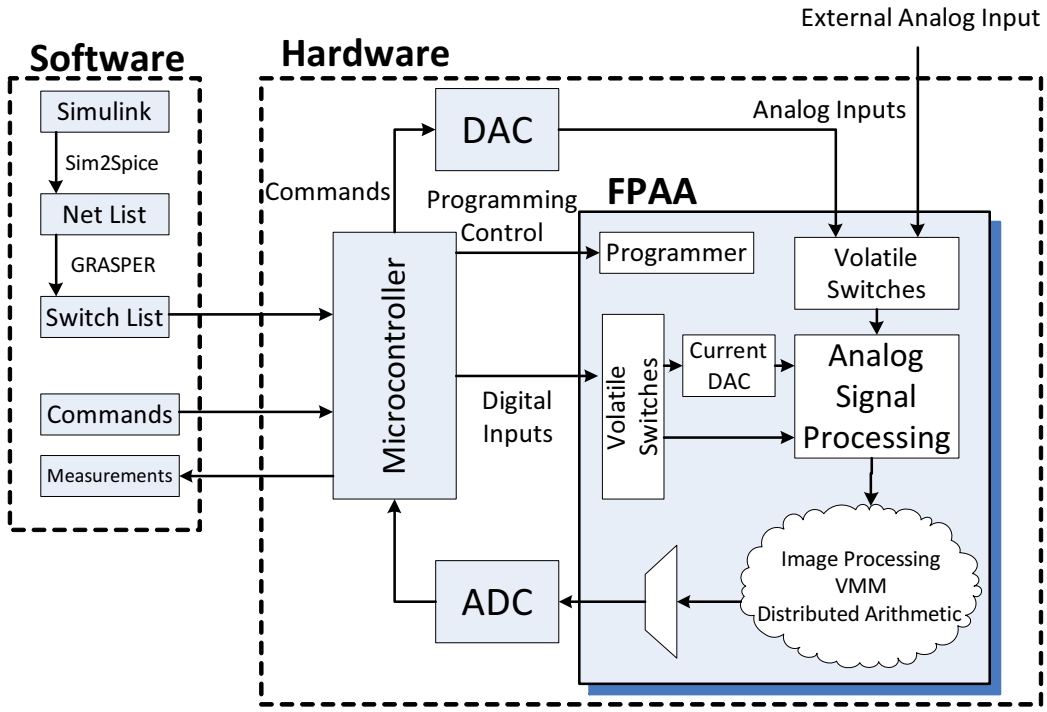


Figure 49: Signal flow in the RASP 2.9v.

microcontroller.

The above features are by now standard for FPAAs developed by the CADSP lab. The RASP 2.9v has a number of major characteristics that distinguish it from previous versions. First, as noted in Fig. 49, digital and analog inputs can be applied to the chip through volatile switches. These switches consist of digital shift registers attached to transmission gates. One side of the transmission gate is tied to a common line, while the other is attached to nets in the switch matrix. This is illustrated in Fig. 50. Inputs can be applied to the switch matrix by activating the transmission gate corresponding to the desired net and applying a signal to the common line. Outputs can be muxed from the switch matrix using these volatile switches.

This chip facilitates communication between the analog and digital signal processing worlds. One heavily used digital operation is vector-matrix multiplication (VMM). The CADSP lab has developed a reliable method for performing VMM using reconfigurable architectures [52]. The RASP 2.9v is specifically designed to facilitate VMM. We added CABs to the 2.9v which are well-suited to compiling VMMs. These CABs have a large number

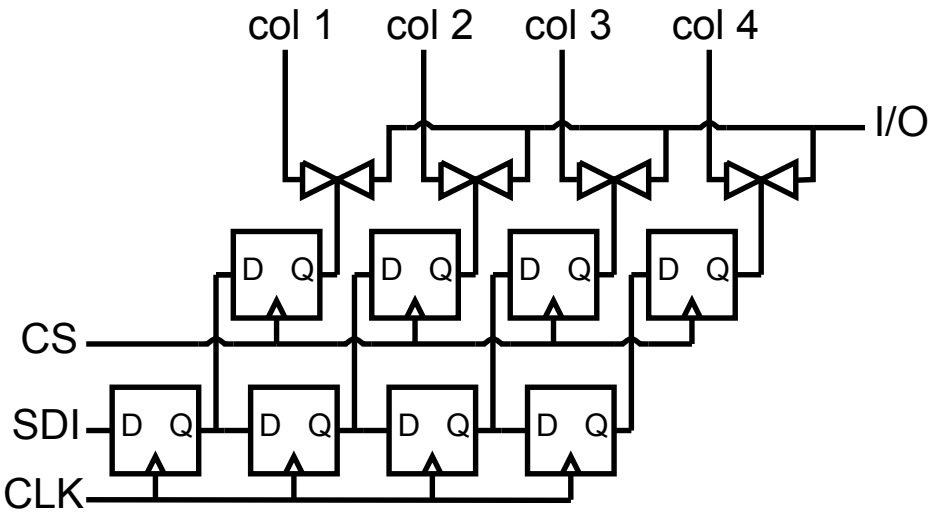


Figure 50: Architecture of the volatile switches in the RASP 2.9v.

of OTAs, and there are local connections which allow few switches to be programmed to create a VMM.

A second important characteristic of the RASP 2.9v which enables VMM is its hybrid switch matrix. Some of its floating-gate elements are programmed using a direct scheme, and others are programmed using an indirect scheme. Direct programming uses one FG-pFET for both programming and in-circuit operations. Indirect programming uses one FG pFET for programming, but a different device with the same gate voltage is used in run-time circuits. During program-mode devices must be isolated using a series pFET switch. In run-mode, this switch cannot pass low voltages. So the floating-gate switch works poorly in directly-programmed structures. In indirect cases, there is no switch isolating the run-mode transistor. This is illustrated in Fig. 51.

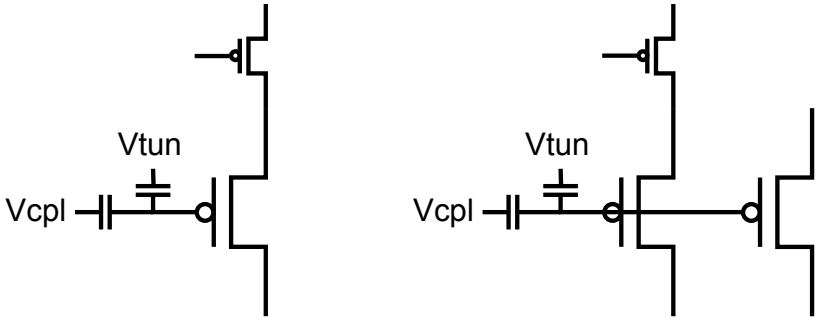


Figure 51: Indirect vs. Direct programming in the RASP 2.9v

The disadvantage of indirect programming is the mismatch between the programmed device and the run-mode devices. This mismatch is a big problem for current-mode circuits such as the VMM. The two devices have the exact same floating-gate voltage, so any mismatch in their physical characteristics (most notably threshold voltage) results in different currents through the two devices for the same set of terminal voltages. Direct programming solves this problem. Current-mode circuits don't need to conduct all the way to ground, so direct programming is ideal since the programmed device is the same device used in run-mode, so mismatch is eliminated.

The other major feature of the RASP 2.9v is that it contains CABs which are well-suited for compiling current-mode DACs. A current-mode DAC is ideal for this chip because the VMMs are also current-mode, meaning that digital signals can be sent from the DAC to the VMM without any intermediate V/I conversions. Details of these DACs are discussed in the following section.

6.2 Design of the DAC CAB

A schematic of the DAC CAB is shown in Fig. 52. The architecture of the CAB is simple. A pFET differential pair steers current between a common output node and ground. The input currents to the pair I_{in_i} come from the switch matrix, external to the CAB. This means that the currents can be generated by an on-chip or off-chip source and routed to the CAB through the switch fabric. However, the CAB was designed in such a way that input currents could be generated from floating-gate pFETs in the switch matrix itself, which makes a compact structure.

The signal which determines whether a particular current is added to the total output or not comes from a serial-in parallel-load shift register. We used the same type of shift register as in the volatile switches. The chip-select signal, which loads data from the input to the output register, is determined by a simple addressing scheme. We have multiplexed the SDI signal so that it can either come from the global SDI line or from a column of the switch matrix. This allows for the DAC to take data from other systems on the chip, increasing the types of systems we can implement with the DAC.

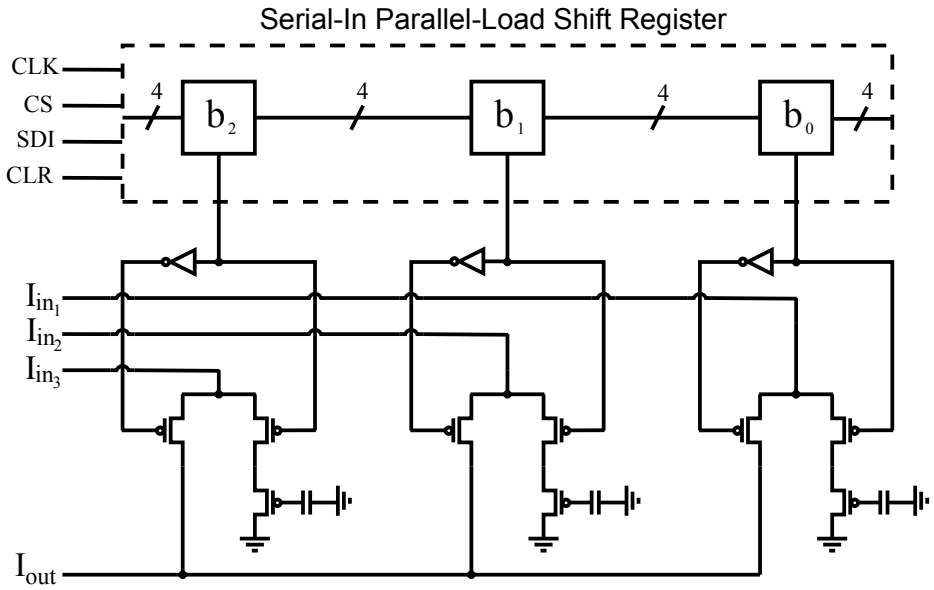


Figure 52: Schematic for three bits of the 8-bit DAC CAB.

While this CAB could have a number of uses due to its reconfigurable nature, it was designed with two particular DAC architectures in mind. The first was a binary-weighted current-steering DAC. A schematic example of this type is shown in Fig. 53. The binary-weighted current sources are floating-gate current sources from the switch matrix. These can be trimmed to precise values with floating-gate programming. This addresses one of the dominant problems in subthreshold current-mode DACs: offset. Any small mismatch between two current sources can have huge effects on the DAC's output. Adding in programmability greatly reduces the offset.

A second type of DAC which can be compiled on this architecture is an R-2R DAC, or diffusor structure. A is shown in Fig. 54. A single floating-gate input current is applied to the first node of the structure. The current is then divided by a proportion set by the floating-gate programming of the longitudinal and leakage pFETs. This current is then used as a binary (or some other proportion) weight for the DAC.

6.3 Experimental Results

The RASP 2.9v was fabricated in a 0.35 μm CMOS process. The resulting die photo is shown in Fig. 55(a), and the DAC CAB portion of the layout is shown in Fig. 55(b). Some specifications for the chip are given in Table 1.

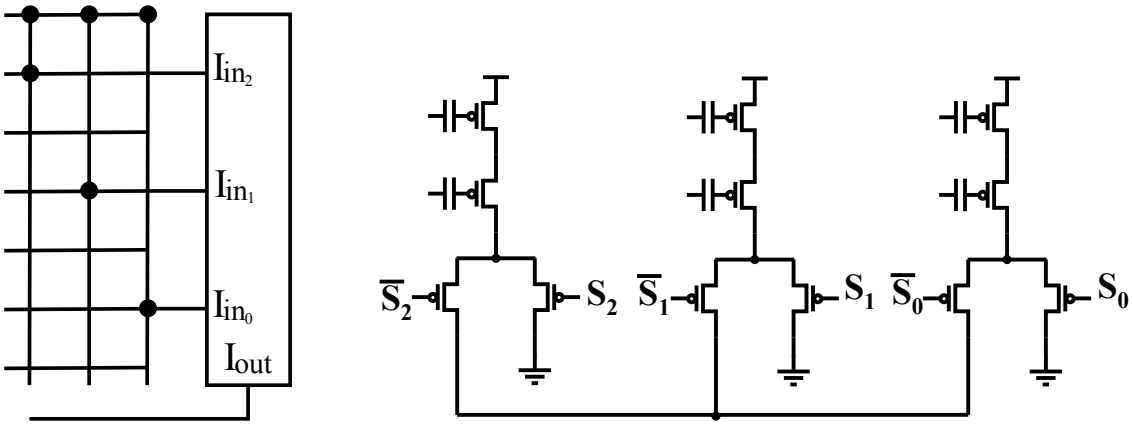


Figure 53: (a) Switch settings for compiling a binary-weighted DAC. (b) Resulting schematic when switches from (a) are programmed.

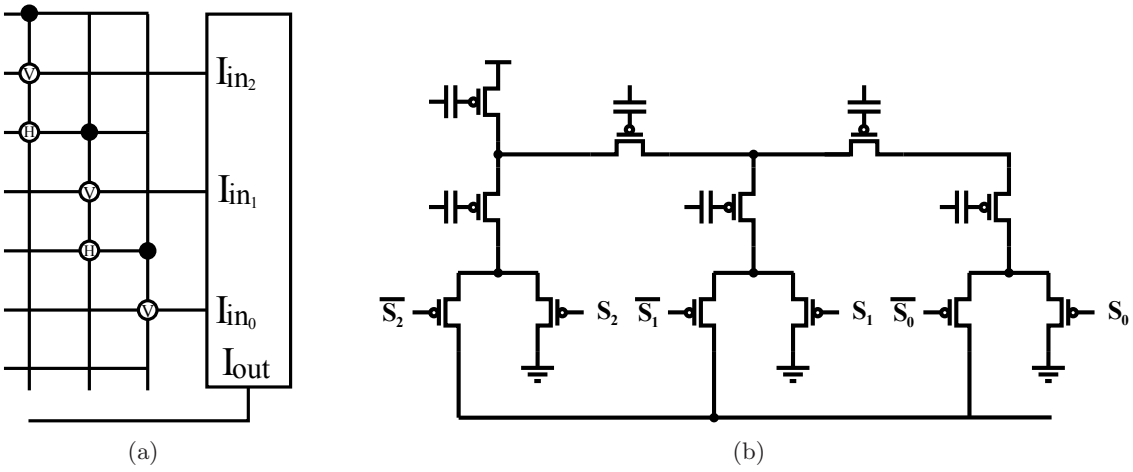
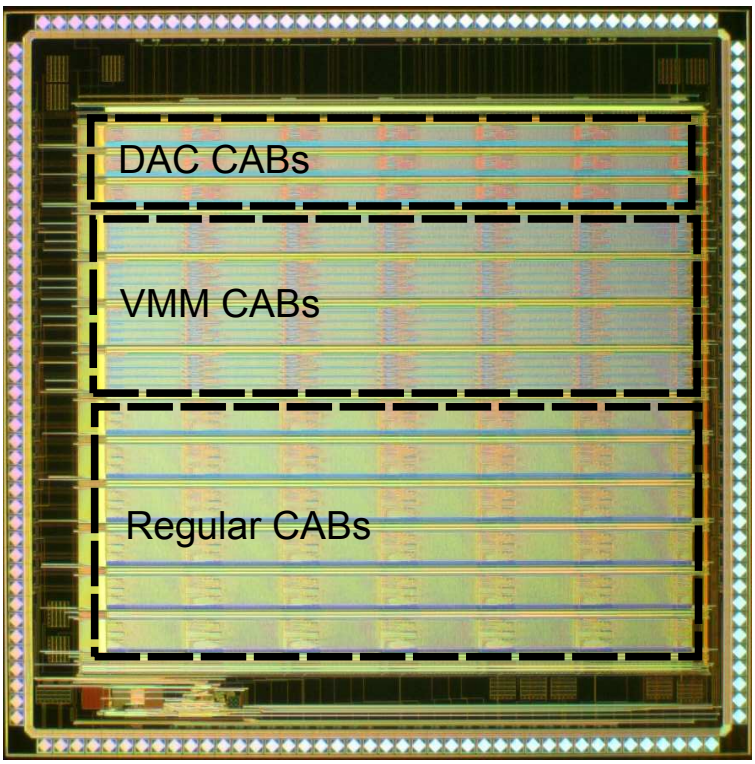


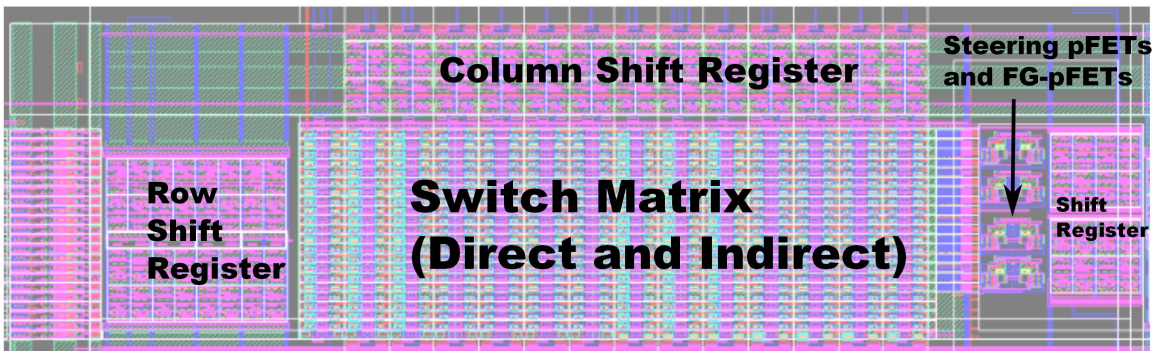
Figure 54: (a) Switch settings for compiling an R2R DAC. (b) Resulting schematic when switches from (a) are programmed.

An 8-bit version of the current-steering DAC was implemented. To achieve higher accuracy, we used a heuristic to trim the floating-gate targets. We give the FPAA initial targets for each of the bits, measure the actual current through the bits, and then modify the targets to get closer to the ideal behavior. This heuristic is shown in pseudocode below.

```
I_ideal = LSB*[1 2 4 8 16 32 64 128] % Goal currents
I_prog = I_ideal                     % Initialize programming targets
for i=1:10 {                         % Ten iterations of heuristic
   eraseFPAA()                       % Tunnel and reverse tunnel
```



(a)



(b)

Figure 55: (a) Die photo of the RASP 2.9v chip. (b) Detailed layout picture of the DAC CAB and associated circuitry implemented in this work.

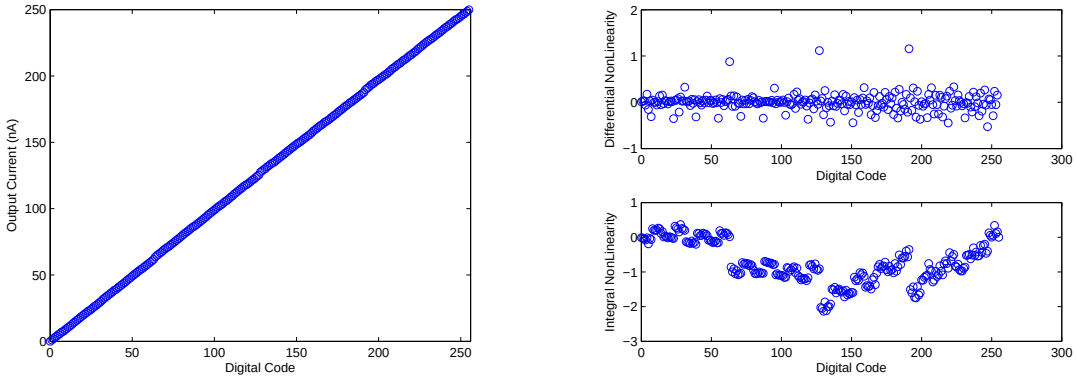
```
programTargets(I_prog)              % Inject targets for this iteration
for j=1:length(I_prog) {            % For each target...
   iout = measureBits(j)            % (1) Measure its current
   ratio = I_prog(j)/iout           % (2) Find correction factor
   I_prog(j) = I_prog(j)*ratio }}   % (3) Create target for next iteration
```

The results of our initial testing with an 8-bit, 1 nA LSB DAC are shown in Fig. 56.

Table 1: RASP 2.9v Specifications

Spec	Value
Process	350nm CMOS
Die Size	5mm x 5mm
CABs	18 DAC, 36 Regular, 24 VMM (x4 input structures)
Chip I/O	79 Analog, 20 Volatile lines
Number of Volatile Switches	4728: 6 x 400-bit (vertical) 14 x 156-bit (horizontal) 6 x 24-bit (DAC)

The measured LSB was 0.978 nA, the worst-case INL was -2.13 LSB, and the worst-case DNL was 1.16 LSB.



(a) (b)

Figure 56: (a) Measured currents for an 8-bit 1-nA LSB DAC (b) DNL and INL for the DAC.

We also compiled an R-2R diffusor structure on the RASP 2.9v. However, the R-2R architecture is much more sensitive to mismatch between floating-gate elements and is difficult to tune because one stage directly depends on the other. Therefore we have not yet achieved results with the R-2R DAC which are as accurate as those from the regular DAC. We hope to improve the tuning heuristic and eventually achieve an R-2R DAC.

Another important part of the DAC CAB architecture is the ability to accept inputs from either the SDI input line or the switch fabric directly. This feature allows for more complex systems to be built, where feedback can be applied from some other subsystem on the chip. We tested this by programming a simple DAC, shifting a series of ones followed

by a series of zeros through the SDI, changing the MUX to allow inputs from the switch fabric, and applying the same set of ones and zeros through a digital input from the switch matrix. We got the same response in both cases, indicating that the DACs can take digital input from the switch fabric. This result is shown in Fig. 57.

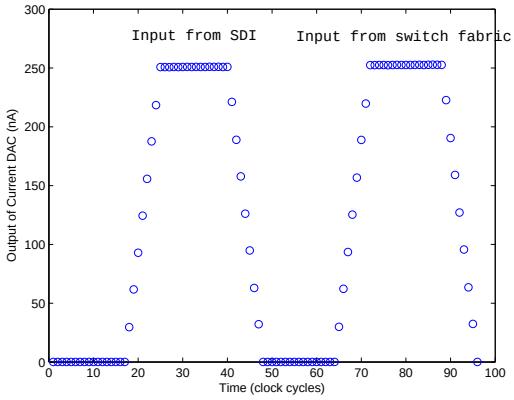


Figure 57: Applying inputs to the DAC both from the SDI input and the switch fabric.

This work has shown that the DAC CAB functions as it was designed for compiling current-steering, binary-weighted DACs. This allowed other work to use the DAC as a building block in more complex systems.

6.4 Application: Image Processing in Current Mode

The purpose of the RASP 2.9v is to enable enhanced digital control of FPAAs for signal-processing applications. To demonstrate the signal-processing capabilities of the FPAA, we performed two image transforms on an image of Tech's mascot, Buzz, as shown in Fig. 58.

Image processing on the 2.9v is accomplished using vector-matrix multiplication. This is accomplished with the circuit shown in Fig. 58a. In this diagram, two input currents are each multiplied by two weights. The two currents represent a signed signal, and the two weights represent a signed weight, so this is a four-quadrant multiplier. The circuit works by translating the input current into a source voltage on the input FG pFET, and this voltage is then translated into a weighted output current by the weighting pFET. Since the two pFETs share a source terminal, the ratio of their currents will be a strong function of the difference in their floating-gate voltages and a weak function of the difference in their

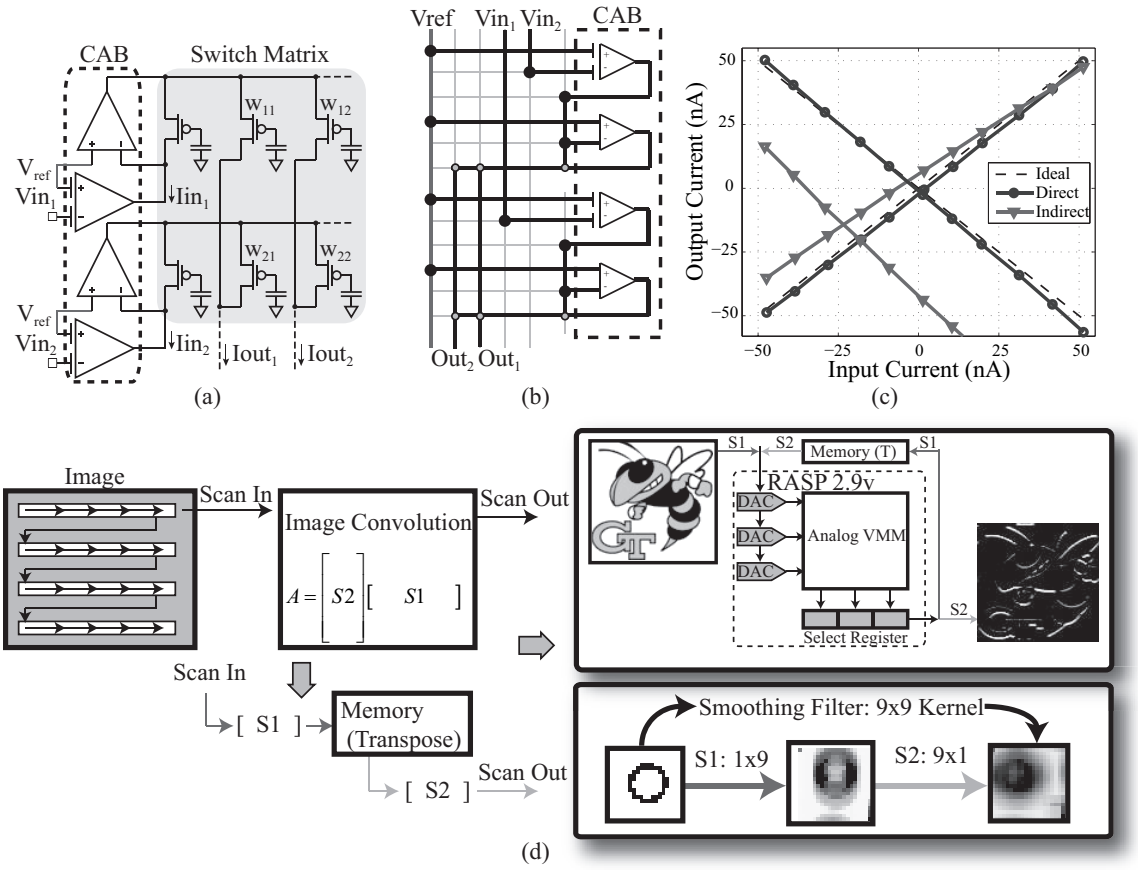


Figure 58: Example image processing application on the RASP 2.9v. (a) Vector-Matrix Multiplication (VMM) is accomplished with pairs of source-coupled floating-gate pFETs operated in their subthreshold regime. (b) VMM is accomplished in a space-efficient manner using a VMM CAB. (c) Comparison of first-pass VMM results between implementations with direct FETs and indirect FETs. (d) Schematic of 2-D convolution method. Image vectors are applied to the chip through the current-mode DACs designed in this work. Their current representation is multiplied with the first half of a separable kernel, and the result is read and stored off chip. The result is transposed and run through a second time with the other half of the decomposed kernel, and the result is presented. We used an edge-detection Sobel kernel and a Gaussian smoothing kernel; image from [53].

drain voltages. If we neglect drain dependence, the multiplication factor can be found by writing down the ratio of the two currents:

$$\begin{aligned}\frac{I_{out}}{I_{in}} &= \frac{I_0 e^{(\kappa(V_{dd}-V_{fgout})-(V_{dd}-V_s)/U_T}}{I_0 e^{(\kappa(V_{dd}-V_{fgin})-(V_{dd}-V_s)/U_T}} \\ &= e^{\kappa(V_{fgin}-V_{fgout})/U_T}. \end{aligned} \tag{39}$$

The current-mode DACs were used to provide input vectors to a Vector-Matrix Multiplication circuit. We performed the transform using separable kernels. This transforms the 2-D convolution operation into two 1-D kernel operations. A separable kernel is a matrix that can be decomposed into the multiplication of two vectors. We chose a Sobel kernel for our application. The convolution with this Sobel kernel decomposes into two 1-D convolutions:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} * \left(\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \right) = \left(\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \right) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}. \tag{40}$$

After we send the inputs into the VMM, the output is measured and stored offchip. We perform a transpose of the matrix and feed the measured values back into the VMM using the other half of the kernel. The results are transformed images. The Sobel kernel performs edge detection, while the smoothing kernel $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ blurs the image.

6.5 Conclusion

The RASP 2.9v was designed in response to the need for more digital control of FPAAs. We designed a current-mode DAC CAB that allow users to send currents into the circuits they are testing. We used these DAC CABs to provide inputs into a VMM-based image processing circuit we compiled on chip. We were able to perform edge detection and smoothing using these DACs and VMMs, demonstrating a useful low-power analog signal processing system.

CHAPTER VII

DESIGN AND LAYOUT OF THE NEXT GENERATION MIXED-SIGNAL FPAAS

So the next generation of FPAAs – the RASP 3.0's – all have a synthesized microcontroller on chip to further enable digital control. We chosea Texas Instruments MSP430 processor because it is widely recognized for its low power dissipation. Rather than obtain IP from TI directly, we used a project on the openCores web site called the openMSP430 [19]. This software package contains a Verilog implementation of the MSP430 which can be targeted to FPGAs or ASICs. We wrote custom peripherals which connected to the MSP430 in order to interface with the circuitry on the FPAA. We discuss the basic 3.0 architecture in Section 7.1.

We first migrated the floating-gate programming control circuitry to take advantage of the on-board MSP430. For the most part, this involved the removal of custom digital circuitry from the programmer subsystem and the insertion of synthesized Verilog peripherals connected to the MSP430. The benefit of doing this is that the programming control functions are easy to change with just a few lines of code. The users now control all of the programmer's functions from assembly or C programs written on the MSP430, whereas previously they were controlled by a plethora of signals originating from an SPI shift register. We added more DACs to the programmer to potentially allow for the programming of floating-gate nFETs. The new programmer is discussed in Section 7.2.

Second, we designed and laid out the successor to the Neuron1D, the RASP 3.0h. This chip contains a number of important upgrades which we deemed necessary based on the testing of the Neuron1D and my goal to have the chip be capable of implementing Independent Component Analysis (ICA) based on Peter Földiàk's work. Since the 3.0H is a member of the 3.0 family, it has an on-chip MSP430 microcontroller as well as custom Verilog peripherals written specifically for the 3.0H. The primary new neuromorphic feature of the chip

is that it adds homeostatic synaptic scaling – a process meant to control the average firing rate of a neuron by multiplicatively scaling the weights of its synaptic inputs. The second important new feature is the addition of inhibitory STDP synapses, which are required for ICA, as well as an increase of the number of synapses to over 200,000 – almost a tenfold increase over the 1D. There is evidence in biology that inhibitory synapses follow STDP rules, although there is a large variation in which specific rule is followed [14]. However, an STDP window similar to the traditional excitatory STDP window has been observed for inhibitory synapses [21]. Finally, we increased the number of neurons on the chip to 256, and each neuron block contains extra channel modules not included in the Neuron1D. The RASP 3.0h is discussed in Section 7.3.

Finally, we designed and laid out subcircuits for the RASP 3.0RF, an FPAA intended to be used at high frequencies. This chip was designed at the 45nm CMOS process node, since reduced parasitics allow higher frequencies of operation. We implemented an OTA-based gyrator circuit used to emulate an LC delay line for use in We also designed a PCB for chip testing. The RF chip is discussed in more detail in Section 7.4.

7.1 The RASP 3.0 Generation of FPAAs

The RASP 3.0, shown schematically in Fig. 59, is a major step forward in the evolution of FPAA design. As with previous designs, the "core" of the RASP 3.0 chips consists of reconfigurable analog primitives that can be connected together using floating-gate switch elements. Different versions of the chips contain different analog processing cores. For example, the RASP 3.0a consists of an FPAADD core [62], while the RASP 3.0H is a Neuron1D core [12].

The main addition to this generation is a synthesized microprocessor, the openMSP430 [19]. As its name implies, the openMSP430 is an open-source hardware project (hosted on opencores.org and authored by Olivier Girard) which emulates the functionality of Texas Instruments' MSP430 microprocessor in Verilog. It is a 16-bit microcontroller that has a watchdog timer, GPIO, and the ability to easily incorporate custom peripherals. The MSP430 enables us to control almost every aspect of the FPAA on chip, conserving power

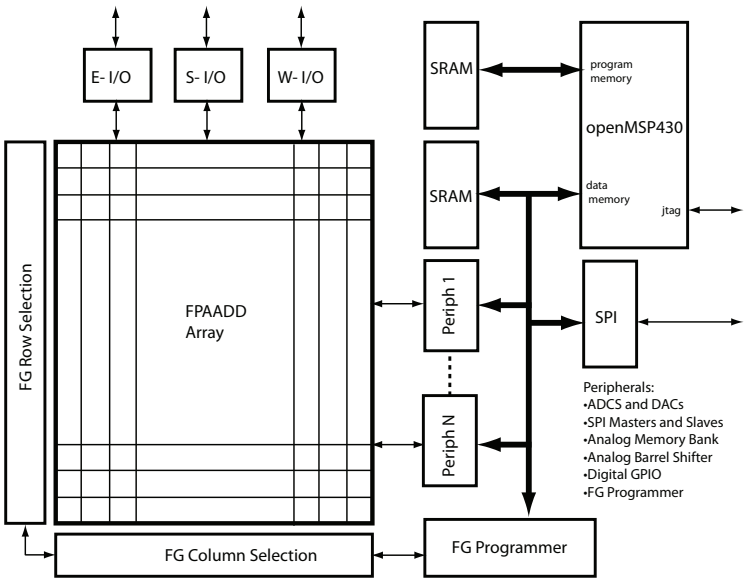


Figure 59: High-level schematic of the RASP 3.0 generation of chips. A synthesized openMSP430 is on chip, connected to SRAM for its program and data memory. The MSP430 is connected to the rest of the FPAA through a digital memory-mapped I/O structure. Our programmer, DACs, ADCs, SPI, GPIO, and analog memory are all connected to the openMSP430 in this way.

and increasing speed. This control is accomplished through a simple memory-mapped I/O interface. Our analog peripherals are connected to registers that correspond to addresses in the MSP430's memory map. When we want to change an analog parameter, we set a memory address to change the peripheral's register, which in turn controls some aspect of the analog behavior.

The on-chip DAC bank is an example of using the memory-mapped I/O to control part of the chip. Our chip has 16 7-bit current-steering DACs, controlled by a total of 8 16-bit registers. Each register bit and its complement is connected to a differential pair in the DAC, therefore steering the current either to the dump or to the output. A schematic of this structure is shown in Fig. 60.

Since different cores require different peripheral functionality, different chips in this family have some peripherals in common, and some are unique to each chip. All chips include a synthesized MSP430, a programmer peripheral, an SPI controller, and some amount of SRAM. We summarize the features of the 3.0 chips in Table 2.

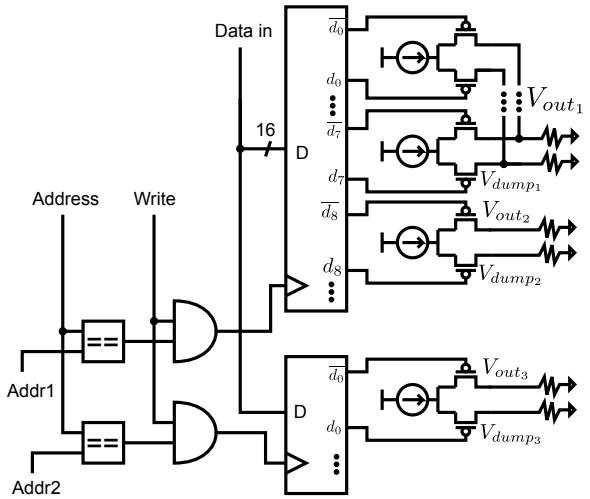


Figure 60: Example of a memory-mapped I/O controlling an analog peripheral. When a DAC value is to be changed, the peripheral address is set to the address of that DAC in the memory map. If the write bit is high, then the DAC's register is updated with whatever is on the Data line. In this example, the digital data then controls a differential pair which sources current either into the output node or a dump node.

Table 2: Summary of the features of each RASP 3.0 chip

Chip	Core	SRAM	DAC	ADC	GPIO	AER	Other
3.0a	FPAADD (14x14)	16 kB	16	2	16b in, 32b out	no	analog memory, barrel shifter
3.0mini	FPAADD (7x14)	16 kB	16	2	16b in, 32b out	no	analog memory, barrel shifter
3.0n	Neuron2D (8x14 A/D, 9x7 neuron)	24 kB	16	2	16b in, 32b out	200 in, 70 out	on-chip biases, 2D control signals
3.0h	Neuron1D (256 neurons, 200k syn)	24 kB	no	no	no	512 in, 256 out	on-chip biases, 1D control signals

7.2 The RASP 3.0 programmer

The basic components of the programmer remain the same as in the RASP 2.8 and 2.9 systems [6], shown schematically in Fig. 61. When a floating-gate is selected for programming, its drain and gate are muxed to connect with the drainline and gateline of the programmer. The programmer muxes the gate and drain potentials to different values depending on the stage of programming. During tunneling, all of the FG terminals are set to ground except for the tunneling junction, which will be pulled high. This places a large voltage across the MOSCAP, causing the SiO2 barrier to change shape, and allowing electrons to tunnel off of the floating gate, increasing the floating gate voltage. Conversely, during reverse tunneling, all potentials are set high except for the tunneling junction, which is set low. This forces a large voltage of the opposite polarity across the MOSCAP, tunneling electrons onto the floating gate, therefore decreasing its voltage. During injection, the source (IVDD) is set to a high potential (usually 6V), while the drain is set to a low potential. The gate is then modulated to cause a near-threshold current to flow through the system. When carriers enter the drain region, they impact-ionize and are injected onto the floating-gate, lowering its voltage. We use floating-gate current as a proxy for the floating-gate voltage during programming to evaluate how much more charge needs to be injected. During measurement, the programmer places IVDD at its run-mode condition (typically 2.5V), sets the gate to its run-mode condition (ground), and connects the drain to a current-to-voltage circuit. The programmer chooses either a diode or a logarithmic transimpedance amplifier to measure the current, depending on the magnitude of the current being measured. The diode is typically used to measure large currents, while the logamp measures small currents.

The new feature of programming on the RASP 3.0 chip is that it is controlled completely through memory-mapped I/O (MMIO) peripherals. Both the DAC inputs and the selection signals for all of the muxes in Fig. 61 come from memory-mapped peripherals, connected to level shifters for logic level compatibility. The counter for the single-slope ADC is on the MSP430 as well. The "done" signal from the programmer signals the counter to cease

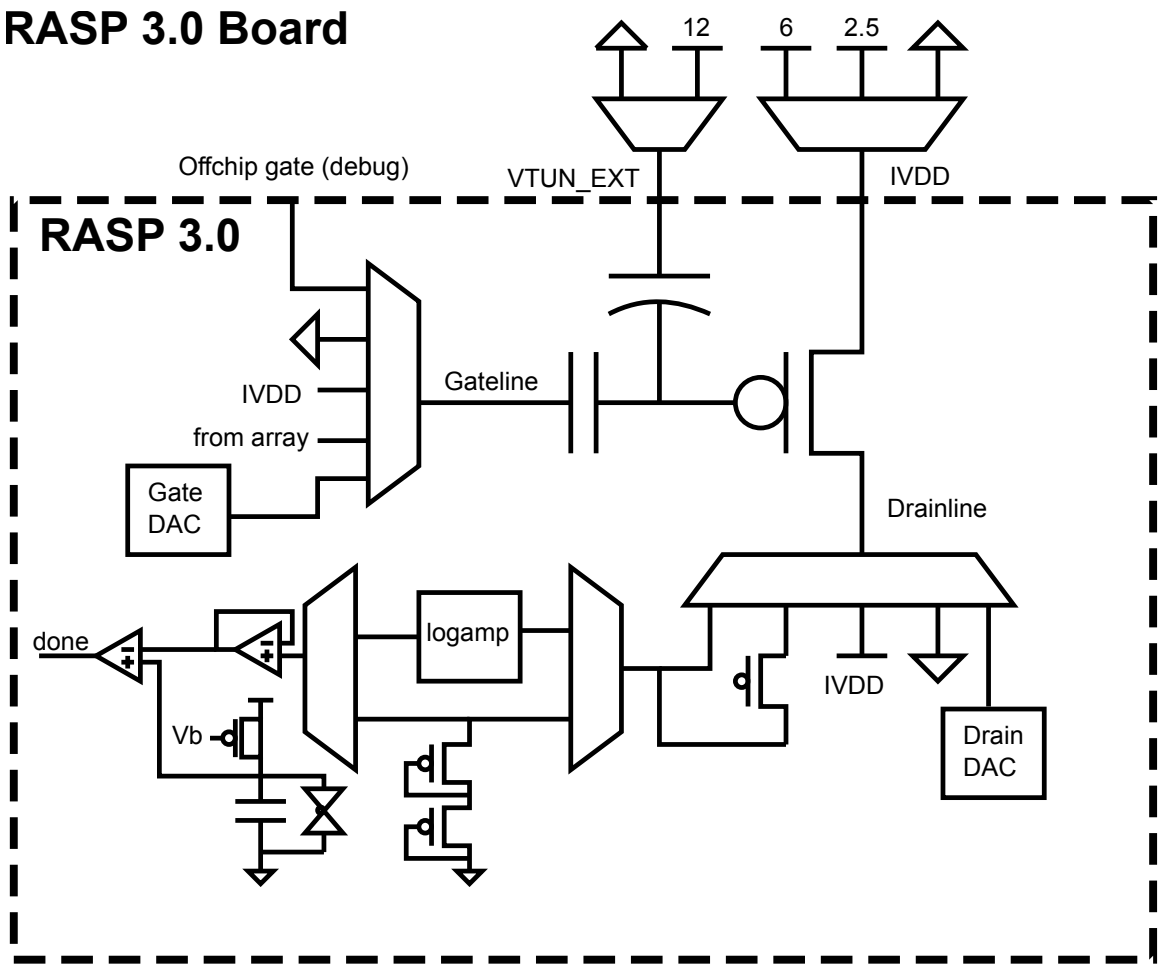


Figure 61: Schematic of the programmer. When a pFET is selected, all of its terminals are accessible. Its source and tunneling terminals are sent off-chip, while its drain and gate terminals are muxed to potentials on-chip. To measure the current through the device, its drain is connected to a current-to-voltage converter (either a diode or a logamp, depending on the situation), and the voltage is fed to a single-slope ADC. The counter for the ADC is in the MSP430, which watches the "done" signal to stop counting. This architecture is mostly unchanged from [6], but it has been reorganized and connected to the openMSP430. All of the select signals for the muxes seen in this figure originate from peripheral registers on the MSP430.

counting, and the value is stored in a register in the MSP430. An entire programming algorithm can be run on the MSP430, contrtolling the measurement, tunneling, and injection entirely on-chip. In Appendix A we include a table that lists the signals controlled by the peripheral registers.

Besides its integration into the MSP430, a few new features have been added to the programmer. As shown in Fig. 61, we now have the option to connect the floating gate being measured to a cascode before connecting to the measurement circuitry. The cascode prevents a large voltage from dropping across the floating-gate, thus preventing injection from occurring. In previous iterations, when the cascode was not available, it was not possible to measure the floating-gate current when IVDD was at injection levels because injection might occur, thus making the measurement invalid. Now that we can measure the current through the floating-gate with IVDD high, there is no need to continually ramp IVDD up and down, which historically has taken a lot of time and is a waste of energy. We have also added two new DACs and control circuitry to the programmer. These control the voltage at the coupling node for the pFETs or nFETs that are not being currently injected or measured.

7.3 The RASP 3.0H: A Neuromorphic Processor with Homeostasis, STDP, and Rich Neural Dynamics

The RASP 3.0H is the next generation of neuromorphic chip from the CADSP lab. It is the successor to the Neuron 1D [12]. The "H" stands for homeostasis, which is a key feature of the chip. This section discusses the design of the chip. It begins with a brief discussion of the various improvements made to the chip, and then provides details about each new subsystem. Next, we discuss the primary motivation for creating this network – the possibility of performing Independent Component Analyisis in an analog spiking neural network.

7.3.1 New Features and Layout

The first of three major additions to the chip is a homeostatic mechanism. Homeostasis is the process by which neurons attempt to regulate their average firing rates. The benefit of

homeostasis is that it forces all neurons to have some average firing rate, encouraging the network to learn weights such that all neurons are used, on average, increasing the likelihood that neurons learn different but important features of their input. Our experiments with the Neuron1D indicated that we needed better control of the total neural excitation. We often observed that a single synapse would start to dominate the behavior of the network, and eventually its weight became so strong that it pulled the membrane voltage constantly high. We chose to implement multiplicative synaptic scaling as our homeostatic mechanism because its implementation as a floating-gate current mirror has a simple and compact circuit realization.

Second, we added inhibitory STDP synapses. These synapses learn according to an STDP rule, but they sink current from the neuron's membrane rather than source current onto the membrane. They are a key feature in networks which perform Independent Component Analysis (ICA). We also increased the number of synapses to over 200,000, which enables significantly more interesting learning experiments than the previous chip, which had just 30,000 synapses.

Finally, we added extra ionic channel models in the neuron blocks. The original blocks only had the sodium and potassium Hopf channels from [17]. This chip includes the sodium and potassium saddle-node channels from [5], providing users more options for the dynamic behavior of the neurons. We increased the number of neuron blocks to 256 (compared to 100 on the Neuron1D), enhancing the capabilities of our networks.

We also added a bank of on-chip floating-gate biases to replace biases that were previously set externally. This makes the test setup for the chip significantly more compact.

We also added an extra feature in the STDP circuitry meant to help solve the problem of synapses that get strengthened too much. We placed a cascode in series with the injection transistor in the STDP setup. If we limit the injection occurring in the synapse, we can limit its weight growth. While homeostasis was the primary method to deal with our weight growth problem, this is a backup system.

Since this is a 3.0-family chip, we included an MSP430 processor. We wrote a custom peripheral for the neuron chip that controls its various debugging signals.

The layout of the RASP 3.0H in a 350nm process is shown in Fig. 62.

7.3.2 Inhibitory STDP Synapses

The previous Neuron1D chip already implemented a a synaptic learning rule known as Spike-Timing-Dependent Plasticity. We added new inhibitory synapses that can learn according to the STDP rule. Their floating-gate potentials can be modified just like the excitatory STDP synapses, and their inhibitory action is accomplished by sinking current from the membrane rather than sourcing current onto the membrane. This change of current direction is accomplished with a current mirror. A schematic of the combined excitatory and inhibitory STDP synapse system is shown in Fig. 63.

Both excitatory and inhibitory synapses are pFETs, and both learn according to an STDP rule. When a presynaptic spike immediately precedes a postsynaptic spike, the pFET is strengthened through injection. Conversely, when a presynaptic spike immediately follows a postsynaptic spike, the pFET is weakened through tunneling.

The purpose of this circuit was to achieve similar behavior to the anti-Hebbian learning rule from [18]. The rule was $\Delta w_{ij} = -\alpha \left(y_i y_j - p^2 \right)$. When a pair of neurons is correlated in biology, the excitatory weight is reduced, and when the pair's activity is anti-correlated, the excitatory weight is increased. When the neurons are correlated in our system, the pFET turns on more, which sinks more current from the membrane, increasing the synapse's inhibition. This reduces the total current onto the target neuron, hopefully achieving a similar effect to the anti-Hebbian synapses of [18]. The αp^2 term in the anti-Hebbian learning rule is a weight decay term. Weight decay can be implemented in the constant tunneling of our homeostatic system.

7.3.3 Implementing Homeostasis with Floating-Gates

The most significant change to this chip is the addition of homeostatic circuitry. We implemented synaptic scaling, which multiplies the neuron's input current by a constant factor determined by the neuron's activity. If the neuron exhibits a low firing rate, its input is multiplied by a constant greater than one to increase its acivity. If it is highly active, its input is multiplied by a constant less than one to decrease its activity. If it is spiking at the

Figure 62: RASP 3.0H chip layout in a 350nm process. Signal flow begins with digital spike generation by AER circuitry or recurrent connections. These signals are sent into circuits which generate triangle waveforms that will then be fed into the synapse array. The synapses then send current down into the neural array, where neurons will integrate the currents onto the soma and generate action potentials. These action potentials are picked up by spike detectors, which feed them into AER circuitry for spike storage and feed them back to recurrent connections in the synapse array. A significant amount of area is taken up by the digital infrastructure – an MSP430 controls the entire chip, along with its associated peripherals and SRAM. A large amount of area is taken up by programmable voltage references, used to set analog biases in the neural circuitry.

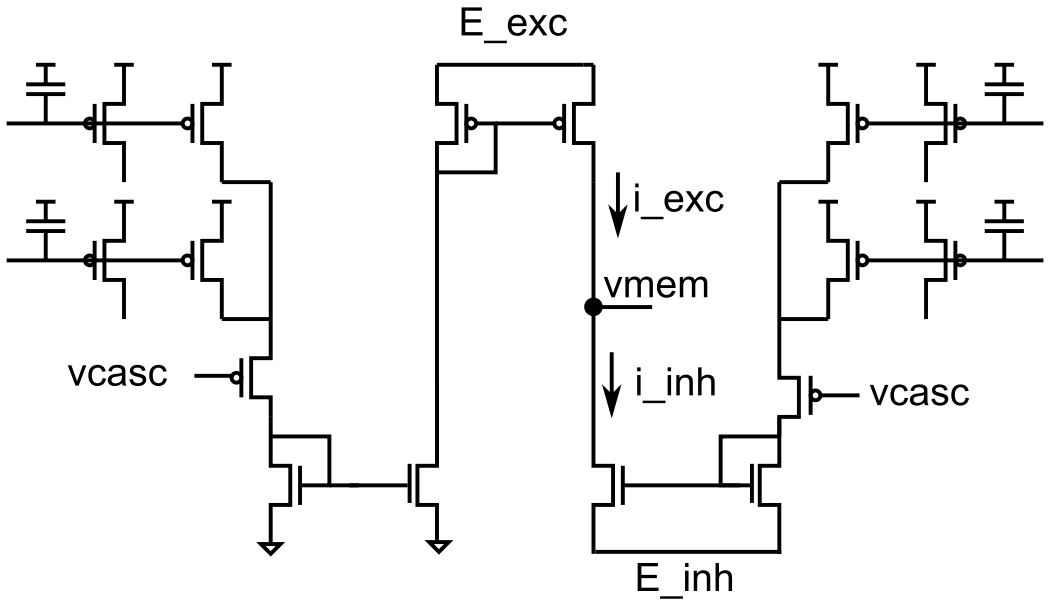


Figure 63: Simplified diagram of STDP-enabled inhibitory synapses

target rate, its input is passed through without any modification.

We initially designed a homeostatic mechanism that we later discovered would not be effective. However, we can fix the problem with a metal-mask change. So the following discusses the circuit that we would design for the metal-mask change.

A schematic of the synaptic scaling mechanism is shown in Fig. 64(a), and a simplified diagram of its operation is shown in Fig. 64(b). The circuit is simply a floating-gate current mirror. The input current and output current are related by a multiplicative constant related to the difference between the trapped charge on the two floating-gates. This constant is expressed as

$$\begin{aligned} I_{in} &= I_0 e^{\frac{\kappa(vdd - v_{fg2}) - (vdd - E_{exc})}{U_T}} \\ I_{out} &= I_0 e^{\frac{\kappa(vdd - v_{fg3}) - (vdd - E_{exc})}{U_T}} \\ \frac{I_{out}}{I_{in}} &= e^{\frac{\kappa(V_{fg2} - V_{fg3})}{U_T}} = e^{\frac{\kappa(Q_2 - Q_3)}{C_T U_T}}, \end{aligned} \tag{41}$$

where Q is the trapped charge on the floating-gate, C_T is the total capacitance seen at the floating gate, U_T is the thermal voltage, κ is the subthreshold slope factor, and I_0 is a collection of physical constants.

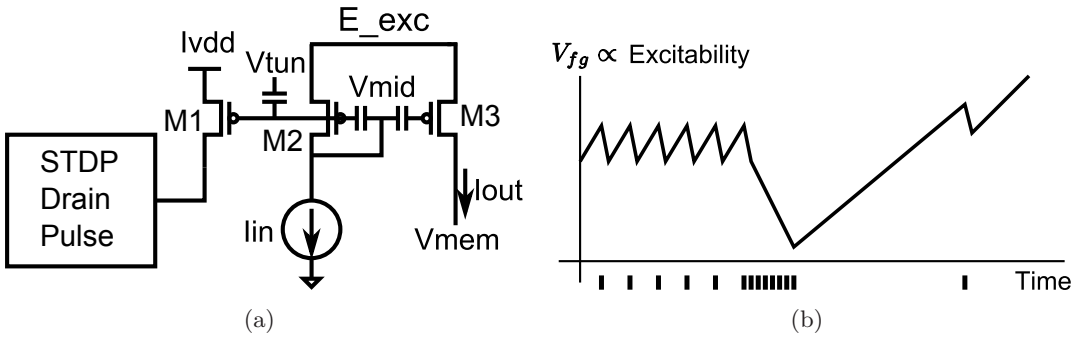


Figure 64: (a) Schematic of homeostasis circuit planned for the metal-mask fix, implemented by a floating-gate current mirror. (b) Example of operation of homeostasis. Output spikes are plotted as short vertical bars beneath the x-axis. Injection occurs whenever an output spike is detected, and this offsets a constant tunneling term. Tunneling increases the floating-gate voltage in the absence of output activity.

The total input current to the neuron is the summation of all the excitatory or inhibitory input current into the neuron. We've implemented synaptic scaling by multiplying the input current by a constant factor which depends on the difference between the charge on the two floating-gates. The charge can be modified only on M1/2 through with tunneling and injection.

We set up a regime of constant, slow tunneling. If the neuron spikes frequently, the drain of M1 will be pulsed down frequently, causing injection to be the dominant driver of O_2. This will make Q_2 decrease, so $Q_2 - Q_3$ decreases, and therefore the output current decreases. So if a neuron spikes frequently, the total input current to it will decrease, causing it to spike less.

Conversely, if the neuron does not spike frequently, the constant slow tunneling will be the dominant effect, raising Q_2 and therefore the output current. The neuron will be more likely to spike in the future. If the parameters are set correctly, we hope that the tunneling and injection will cancel out over time, and the neuron will spike at approximately the target firing rate. All three of these scenarios are depicted in Fig. 64(b).

7.3.4 New and Modified Neuron Channel Models

The RASP 3.0H chip includes improved circuitry for modeling the dynamics of neurons. As discussed in [17], we model the ionic channels of biology using subthreshold transistors.

Like ionic channels, transistors in this regime are voltage-gated conductances operating on the principle of diffusion. The challenge of designing neuromorphic circuits is to create circuits that modulate the gate voltage of the transistor channel as a function of time and the membrane voltage such that they emulate the behavior of neurons.

The original channels designed by our lab [17] used a bandpass filter to emulate the gating function of sodium (Na) channels and a lowpass filter to emulate the gating function of potassium (K) channels. The dynamics of these circuits were analyzed in detail in [7], and they were shown to undergo a Hopf bifurcation. We used these channels in the Neuron 1D, and we discovered that a few modifications should be made to improve their performance. First, the Na channel's lowpass corner was too low. Its step response took long to settle back to its resting state. So we removed the feedback capacitor C_z in the Na amplifier. Secondly, the K channel relied on capacitive coupling to achieve its lowpass behavior. When the membrane (the channel's source) was stepped suddenly, this change instantaneously coupled capacitively into the gate node. Ideally, the current through the Na channel shouldn't change if both of these steps are the same magnitude. However, when the total capacitance at the node is large relative to the coupling capacitance, the gate doesn't change as much as the source, and there is an instantaneous change in current. We decided to use a source follower as the gating mechanism and an nFET to act as the channel conductance. These changes are shown in Figs. 65(a) and 65(b). To verify that these changes would work, we ran simulations of the circuit. The results are shown in Fig. 7.3.4

We added two new Na and K channels to the neuron block. Different applications will often require different neural dynamics to implement the network efficiently. Neuron behaviors can be divided into two classes: resonators and integrators. The Hopf neurons are resonators, typically responding to a band of input frequencies. Integrators have a low-pass behavior, which is desirable in some applications. These neurons typically display saddle-node bifurcations. Our lab designed two channels that have been shown to undergo these saddle-node bifurcations [5]. These channels have been added to the neuron block, and they are shown schematically in Figs. 67(a) and 67(b).

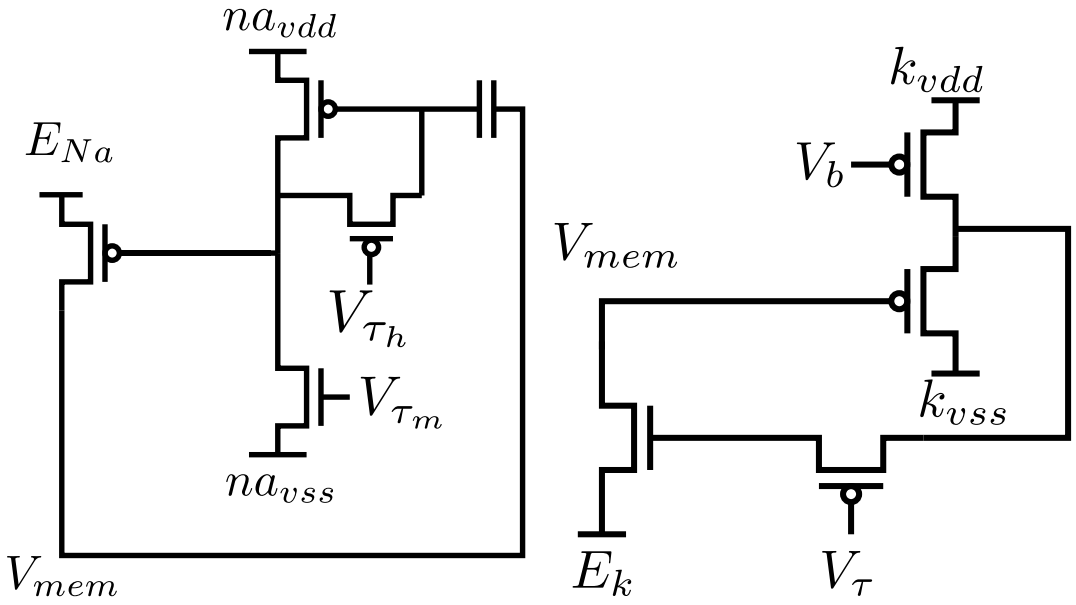


Figure 65: (a) Modified sodium channel of the Hopf neuron. The feedback capacitor of [17] has been removed. (b) Modified potassium channel of the Hopf neuron. The low pass filter now consists of a source follower and an nFET channel.

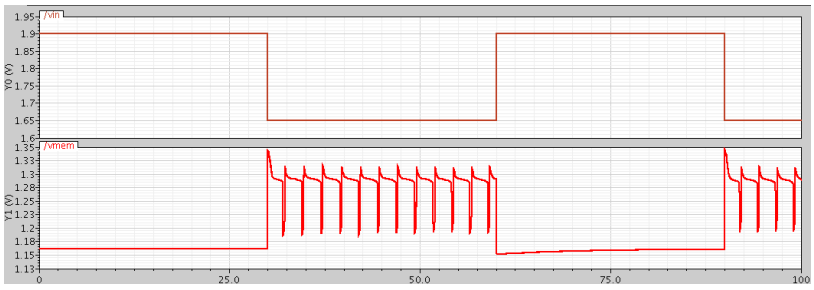


Figure 66: Transient simulation of the new Hopf neuron. The top trace shows the input voltage to a pFET synapse, and the bottom trace shows the neuron's membrane voltage in response.

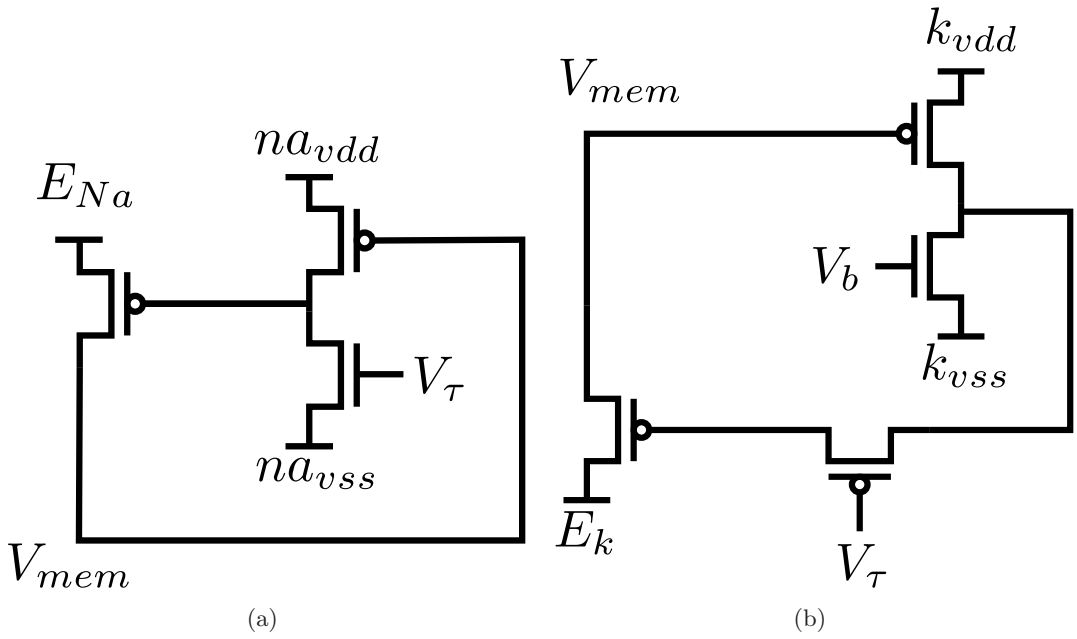


Figure 67: (a) The sodium channel of a saddle-node neuron. (b) The potassium channel of a saddle-node neuron.

7.3.5 Weight-Limiting Cascode

One problem we observed with the Neuron1D chip was the tendency for runaway increase of synaptic weights. As synapses become potentiated, they become more likely to participate in successful firings of their postsynaptic neurons, causing further potentiation. One way to deal with this is to use homeostasis to regulate the average firing rate of neurons. We have included another method to prevent weight growth. A cascode was included in the path of the injection FET of STDP synapses. This cascode limits the injection current into the learning synapse, which should prevent growth of the weight without bound. This cascode also gives us the ability to continuously modulate another parameter of learning, possibly giving more flexibility in designing the learning rule.

7.3.6 Programmable On-Chip Biases

The Neuron1D chip used a large number of analog biases, all of which were provided off-chip from a PCB. We added a on-chip floating-gate bias circuits to the RASP 3.0H to reduce the need for off-chip biasing circuitry. The biases are floating-gate OTA followers, which were also used for biasing the subtractive synthesizer (Fig. 24(b)). Some of the biases needed to

be at voltage levels above run-time IVDD. We designed an op-amp and configured it in a non-inverting amplifying configuration on a higher power supply so that the floating-gate circuit's output is multiplied by a constant gain that enables outputs higher than IVDD.

7.3.7 Target Application: The Földàk Network and Spiking ICA

An important neural network was invented in the early 1990s by Peter Földiàk [18]. The network performs unsupervised learning on its inputs to learn the most efficient way to represent them. One can model an image as the superposition of many simple features, and this network learns those features. This learning allows the network to form a sparse representation of the input in which few neurons are active at once to represent a complex scene. This ability is useful for power-constrained systems.

The network is a simple one-layer network. The inputs are fully connected to the first layer by excitatory synapses which follow a Hebbian learning rule. The Hebbian rule [27] finds and enhances correlations in a network by increasing the weight of a synapse in proportion to the product of its pre- and post-synaptic neurons. If two neurons are active at the same time, their synaptic connection will be strengthened. A pure Hebbian rule is unstable since the weight will grow without bound, so a weight decay term is included. The rule Földiàk used was

$$\Delta q_{ij} = \beta y_i \left(x_j - q_{ij} \right), \tag{42}$$

where the first term is the Hebbian learning term and the second is weight decay.

Földiàk also included an anti-Hebbian learning rule. This causes the excitatory connection between neurons to decrease if they are active at the same time, which decreases correlation between the two. This competitive learning causes the two neurons to compete to represent the same input; eventually one will "win," and the other neuron will learn to represent some other pattern. The rule Földiàk used for anti-Hebbian learning was

$$\Delta w_{ij} = -\alpha \left(y_i y_j - p^2 \right). \tag{43}$$

The final feature in Földiàk's network is a homeostatic mechanism called threshold modification. Each neuron in the network has a firing threshold associated with it, and

this threshold can be modified based on an adaptation rule. Földiàk wanted the neurons self-regulate such that their probability of firing was near constant. He lowered the firing threshold (made the neuron easier to fire) when the neuron was inactive, and he raised the threshold (made it easier to fire) when the neuron was too active. This regulated the activity of individual neurons so that the overall network activity was sparse and evenly distributed among the units. The rule Földiàk used for threshold modification was

$$\Delta t_i = \gamma \left(y_i - p \right). \tag{44}$$

The network was able to perform some interesting tasks. A classic experiment is the Földiàk bar experiment, where the network is presented with visual stimuli that consists of a combination of vertical and horizontal black bars on a white background. After a number of iterations, the network would learn what fundamental visual "features" comprised the input. The network learned to represent the input with vertical and horizontal bars **??**. A similar experiment run on alphanumeric characters yielded character-looking primitives.

7.3.8 Errors and Their Potential Fixes

In the process of checking schematics and documenting the circuitry, we found two schematic errors that should be fixed. The drain of M1 in the homeostasis circuitry was set at a fixed potential, rather than connected to the drain pulse. We have surveyed the layout, and we should be able to connect the drain of each synaptic scaling circuit to the drain pulse line from the neuron. A metal mask change could fix the issue.

Secondly, a schematic error led to the exclusion of an edge detection and pulse generation circuit in the synapses whose input comes from the AER inputs. Recurrent connections still have their pulse generation circuitry. One solution that would allow for the study of small networks is to program the weight of an AER synapse so much that it always causes a recurrently-connected neuron to spike. This recurrent neuron could then act as a proxy for this AER input and time the pulse correctly.

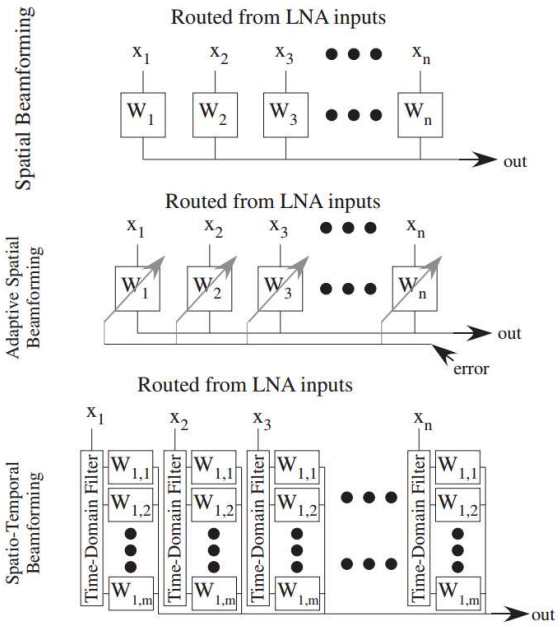


Figure 68: The goal of the RASP 3.0 RF's frontend is to perform spatial and temporal beamforming on the input signal. The weighting functions are performed by vector-matrix multipliers, and the time-domain filter is a delay line implemented with an active inductor and parasitic capacitance of the routing lines.

7.4 The RASP 3.0RF: A High-Frequency FPAA

The RASP 3.0RF is an FPAA designed for RF applications in mind. This chip has been fabricated in a TSMC 40nm process.

The chip consists of two parts – the regular RASP 3.0 structure as discussed earlier, and an RF frontend. This is illustrated in Fig. 7.4. The purpose of the RF frontend is to amplify the RF signal and preprocess it before downconversion. We hope to perform spatial and temporal beamforming on the input signals, as shown in Fig. 7.4.

The architecture of the frontend is shown in Fig. 7.4. The input to the system, on the left, is one or more RF signals. This signal is amplified by a Low-Noise Amplifier (LNA), and then sent into high-speed RF switch fabric. We have designed the RF switch fabric such that it contains active LC circuits. This is accomplished by the addition of an active inductor circuit in the S-block. We form a delay line by routing a signal through a series of S-blocks.

Since the floating gates in the C-blocks are often used as VMM structures, we can create

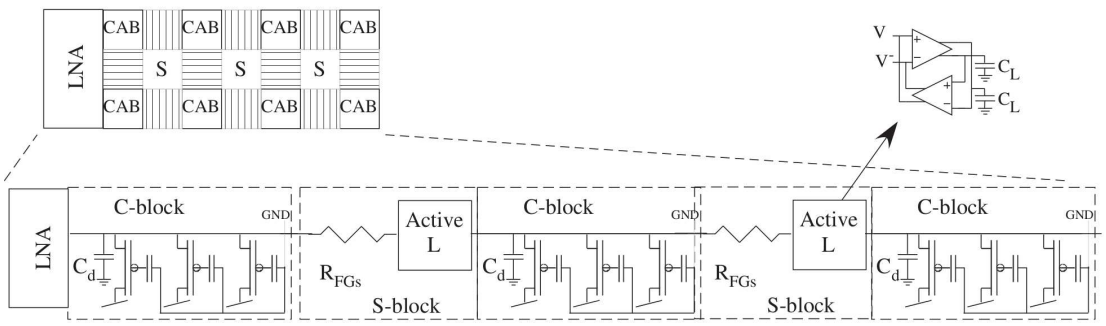


Figure 69: Basic architecture of the RF FPAA. An LNA amplifies an RF input signal. This is fed into a C-block, which contains vector-matrix multipliers. The signal is then routed and delayed using an active-L filter through the S-block. It is multiplied by a weighting function again, and the process is repeated.

an FIR filter using the S-blocks and the RF switch fabric. Such a filter could be used for image rejection in a reciever architecture. This architecture can also perform spatiotemporal beamforming. The RF CABs consist of blocks that are useful in RF signal processing, such as mixers and OTAs. Once RF signal processing is completed, the signal is downconverted inside the RF CABs and sent to the regular RASP 3.0 structure.

The completed system layout is shown in Fig. 7.4.

We have implemented an active-L circuit, which is a critical part of the delay line. The active-L circuit consists of two differential difference operational transconductance amplifiers (OTAs) connected in a feedback configuration. When this structure is repeated and routing capacitance is added in between nodes, a delay line is formed. A schematic of the delay line circuit is shown in Fig. 7.4.

The equivalence between this delay line and an LC circuit is best shown by translating it to a single-ended version, as shown in Fig. 72(a). Writing Kirchhoff's Current Law (KCL) at V_2, we obtain a differential equation describing the LC delay line:

$$\begin{aligned} 0 &= I_1 + I_2 + I_3 = \dot{I}_1 + \dot{I}_2 + \dot{I}_3 \\ 0 &= \frac{(V_2 - V_0)}{L} + C\ddot{V}_2 + \frac{(V_2 - V_4)}{L} \\ CL\ddot{V}_2 &= V_0 - 2V_2 + V_4. \end{aligned} \tag{45}$$

Next we derive the dynamics of the Gm-C delay line. We model the OTAs as ideal, with

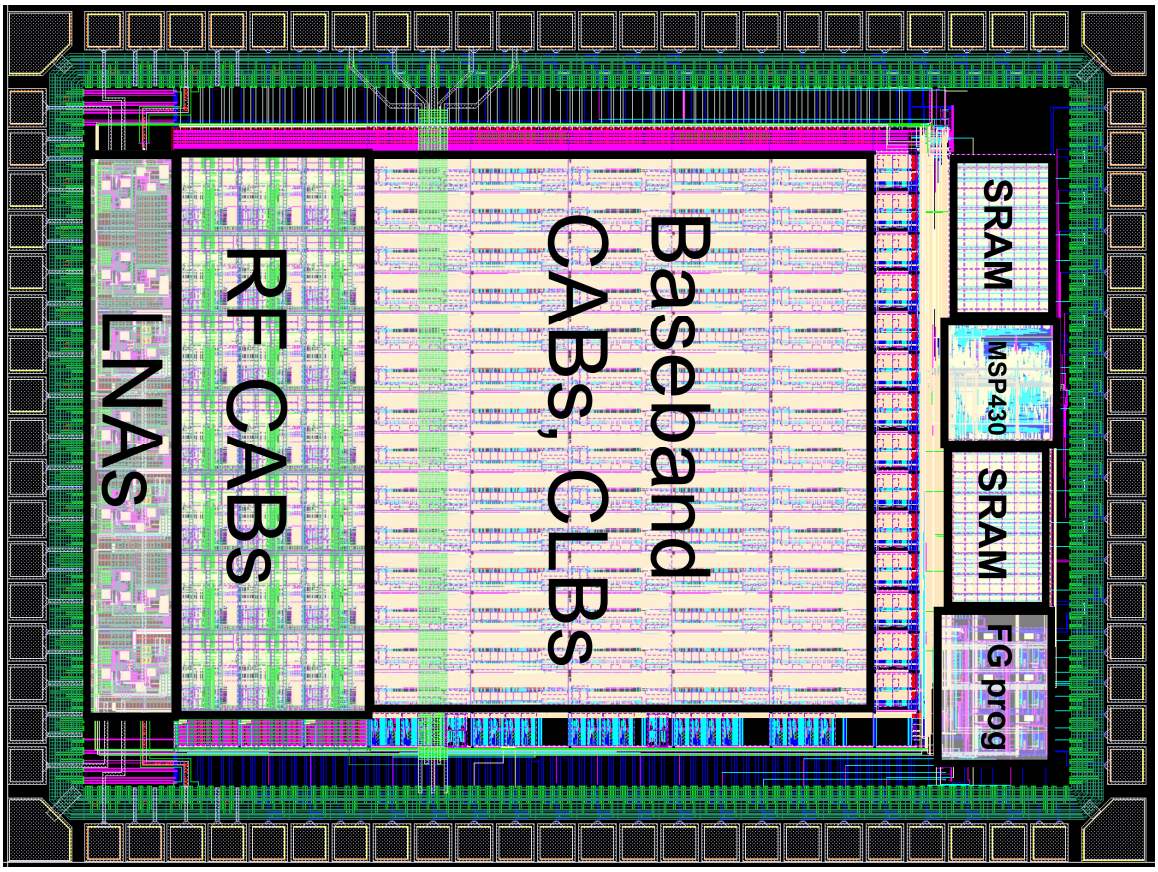


Figure 70: Layout of the RASP 3.0RF chip.

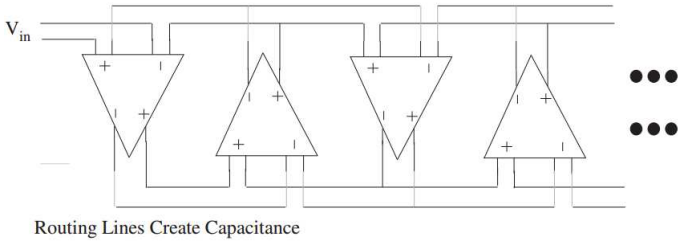


Figure 71: Schematic of the delay line. An active-L circuit is formed by two differential difference OTAs in feedback. When routing capacitance is added to the circuit, a delay line is formed.

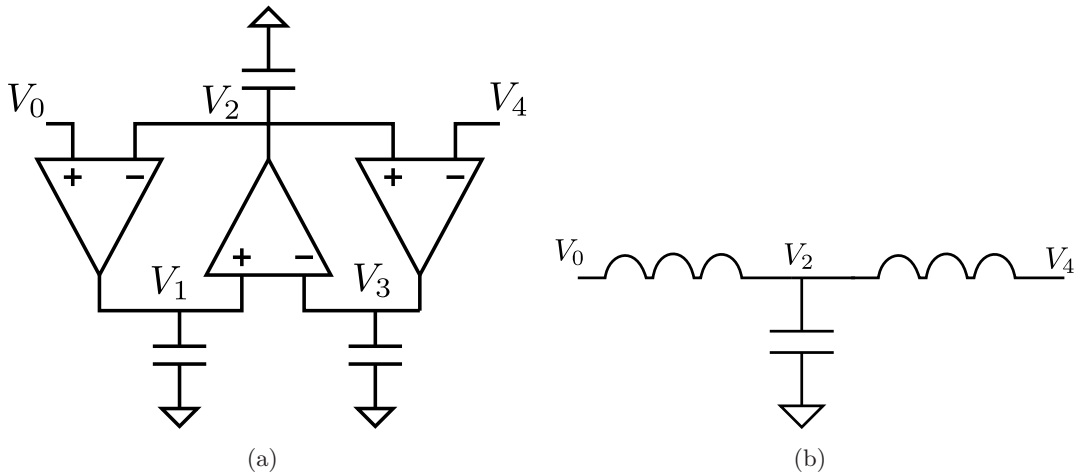


Figure 72: Circuits for deriving equivalence between Gm-C delay line and LC delay line

$I = G_m (V_p - V_m)$. Writing KCL at V_1, V_2, and V_3, we get

$$C\dot{V_1} = G_m (V_0 - V_2) \tag{46}$$

$$C\dot{V_2} = G_m (V_1 - V_3) \tag{47}$$

$$C\dot{V_3} = G_m (V_2 - V_4) . \tag{48}$$

Taking the derivative of Eq. 47 and substituting for $\dot{V_1}$ and $\dot{V_3}$, we get

$$C\ddot{V_2} = G_m \left(\dot{V_1} - \dot{V_3}\right)$$

$$\frac{C}{G_m^2}\ddot{V_2} = V_0 - 2V_2 + V_4. \tag{49}$$

So the two delay lines have the same dynamic behavior.

7.4.1 Differential Difference Amplifier

The delay line is designed to work at 1-4 GHz, so the major challenge was designing an OTA-based differential difference amplifier that works at those frequencies. A schematic of the amplifier is shown in Fig. 7.4.1.

We must include common-mode feedback (CMFB) to ensure that the output voltages remain at a desired bias point. CMFB is performed by the coupling capacitors between the pFET bias node and the output node. The output voltage couples into the bias node, changing the bias current until it matches the current through the nFET bias. More quantitatively, neglecting drain effects,

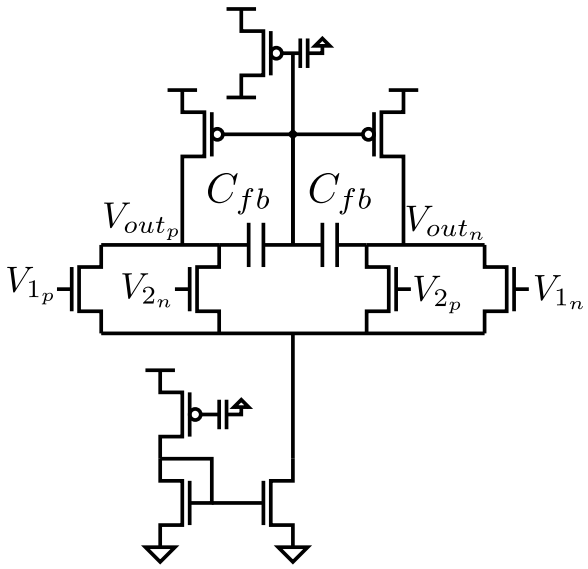


Figure 73: Schematic of the differential difference amplifier. The nFET and pFET biases are both set with floating-gate current sources. Capacitive common-mode feedback sets the DC output voltages as a function of the difference between those current sources.

$$I_{bias_n} = I_{ref_n} e^{Q_n/C_T} \tag{50}$$

$$I_{bias_p} = I_{ref_p} e^{Q_p/C_T - \frac{\kappa C_{fb}}{C_T} V_{out}}. \tag{51}$$

Performing a half-circuit analysis and writing Kirchhoff's Current Law (KCL) at an output node,

$$\begin{aligned} I_{ref_n} e^{Q_n/C_T} &= I_{ref_p} e^{Q_p/C_T - \frac{\kappa C_{fb}}{C_T} V_{out}} \\ V_{out} &= \frac{(Q_p - Q_n)}{\kappa C_{fb}} - \ln\left(\frac{I_{ref_p}}{I_{ref_n}}\right). \end{aligned} \tag{52}$$

The common-mode output voltage is set by the difference between the charge on the two floating-gate biases, which can be programmed quite accurately. We next perform a low-frequency small-signal analysis to demonstrate that this behaves as a differential difference amplifier. Writing KCL at the output nodes and assuming all transconductances are the

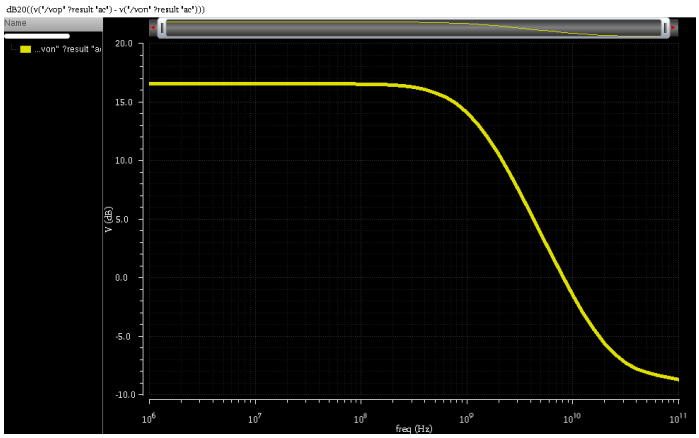


Figure 74: AC response of the floating-gate based differential difference amplifier.

same,

$$i_{out_p} = g_m \left(V_{1_p} + V_{2_n}\right)$$

$$i_{out_n} = g_m \left(V_{1_n} + V_{2_4}\right)$$

$$i_{out_p} - i_{out_n} = g_m \left(\left(V_{1_p} - V_{1_n}\right) - \left(V_{2_p} - V_{2_n}\right)\right). \tag{53}$$

The transfer function verifies that this circuit amplifies the difference between two differential voltages. The simulated frequency response of this amplifier is shown in Fig. 7.4.1.

7.4.2 Delay Line Simulations

We simulated an 8-stage delay line, and the AC response and transient response are shown in Figs. 7.4.2 and 7.4.2, respectively.

7.4.3 Mixed-Signal Programmer Design and Simulations

We revised the high-level programmer design and completed the low-level circuit design. The architecture of the programmer is the same as in the RASP 3.0. The primary new addition is a new DAC output stage. The schematic for the DAC is shown in Fig. 7.4.3. The op amp pictured in the DAC schematic is shown in Fig. 7.4.3.

To verify that the programmer works as expected, we wrote a Verilog testbench and performed mixed-signal simulations using Cadence's AMS.

We first swept the digital codes of all the DACs. The results are shown in Fig. 79(a). Next, we checked the drain selection circuitry. We changed the mux's selection to make sure

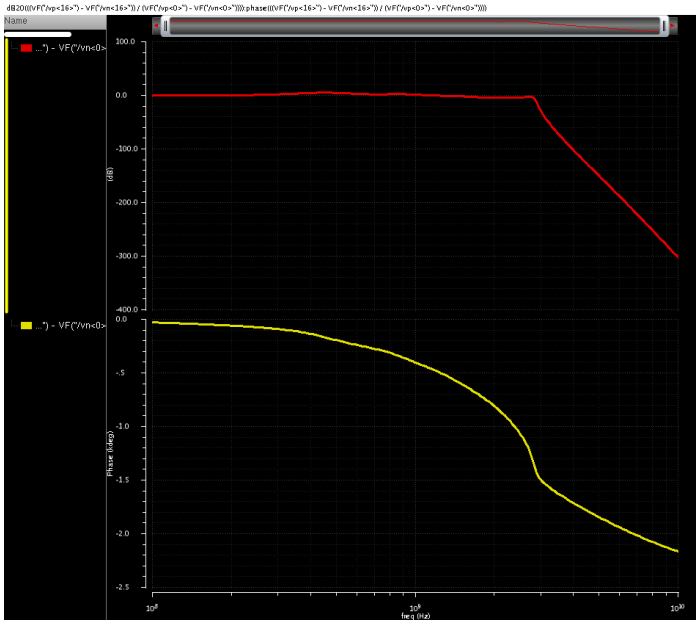


Figure 75: AC response of a 8-stage delay line.

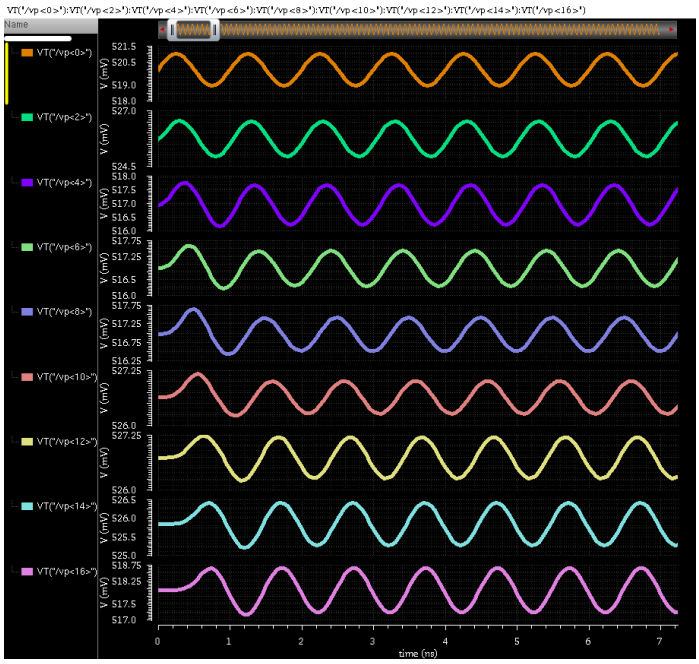


Figure 76: Transient response of a 8-stage delay line to a 1 GHz input sinusoid.

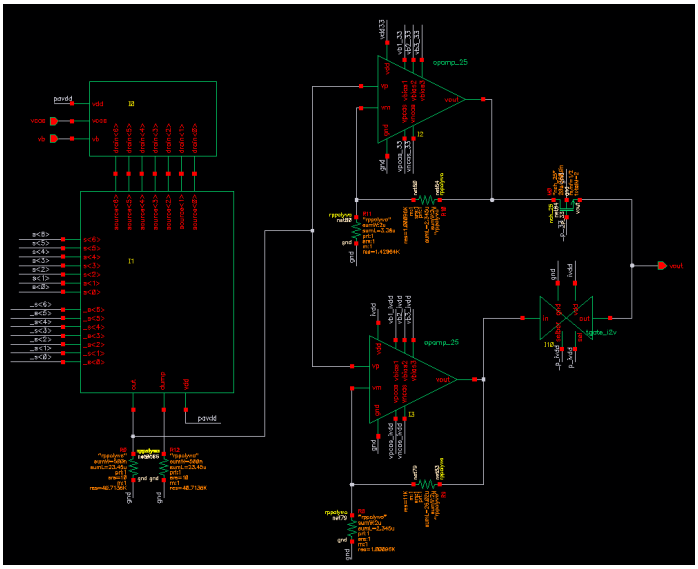


Figure 77: Schematic of the programmer DAC in the RASP 3.0RF chip.

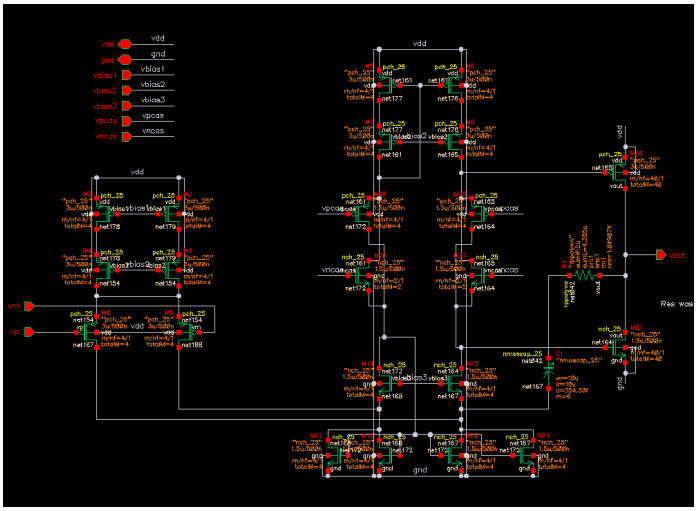


Figure 78: Schematic of the programmer DAC in the RASP 3.0RF chip.

the drain would switch between the various supplies. This is shown in Fig. 79(b). Similarly, the tests for the gate selection are shown in Fig. 79(c). Finally, we tested the drive strength of the DACs to ensure that they would be able to set the terminals correctly. This is shown in Fig. 79(d).

7.5 Testing the RASP 3.0RF

7.5.1 Printed Circuit Board Test Platform

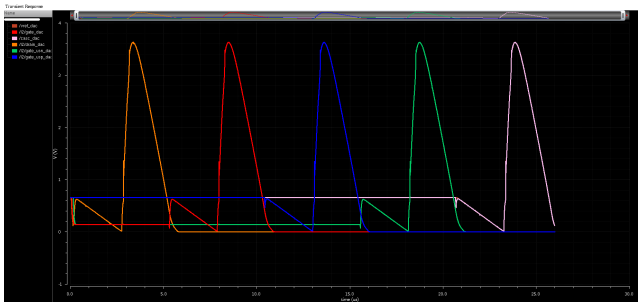
To test the 3.0 RF chip, we designed a Printed Circuit Board (PCB) to serve as a test platform. An annotated picture of the PCB is shown in Fig. 80.

The openMSP430 uses an 8N1 serial interface to communicate with the outside world. An FTDI FT2232C USB-to-serial chip mediates the communication between PC and the MSP430. The other channel of the FT2232C sends general-purpose digital signals to MSP430 – such as debug enable, reset, SPI signals, etc. It was also used to program the clock generator, which generates a clock between 2 and 70 MHz. Inputs to the RF frontend were provided by a 50-ohm microstrip, designed in collaboration with the GEMS lab at Georgia Tech. The local oscillator (LO) can be provided by one of two sources, and the PCB switches between these two sources with a Mini-Circuits SPDT switch. The first LO source is a connection to an SMA connector from an off-board source. The second source is a TI LMX2581 chip.

7.5.2 Testing and Analysis of the Synthesized openMSP430

After verifying the majority of the PCB's functionality, we began testing the digital portions of the chip. Programming of the openMSP430 is controlled by a serial (8N1) debug interface. Users communicate with the chip by reading and writing to registers controlled through the serial interface. We set the chip's baud rate by sending an initialization byte (0x80) to the chip, which starts an on-chip routine that counts the length of the transmission and sets an internal counter. After initalization, we read and write to the debug registers. We show an example of this process on the RASP 3.0 FPAA in Fig. 81.

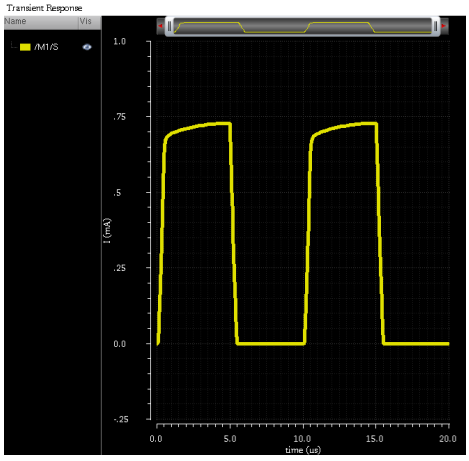
Unfortunately, all our attempts to perform the same measurement on the RASP 3.0RF were unsuccessful. We observed a wide range of behaviors. Some chips would always return



(a)

(b)

(c)



(d)

Figure 79: Testing the 3.0 RF chip's programmer DACs. (a) Testing the DACs. (b) Testing drain selection. (c) Testing gate selection. (d) Testing the drive strength of the DACs.

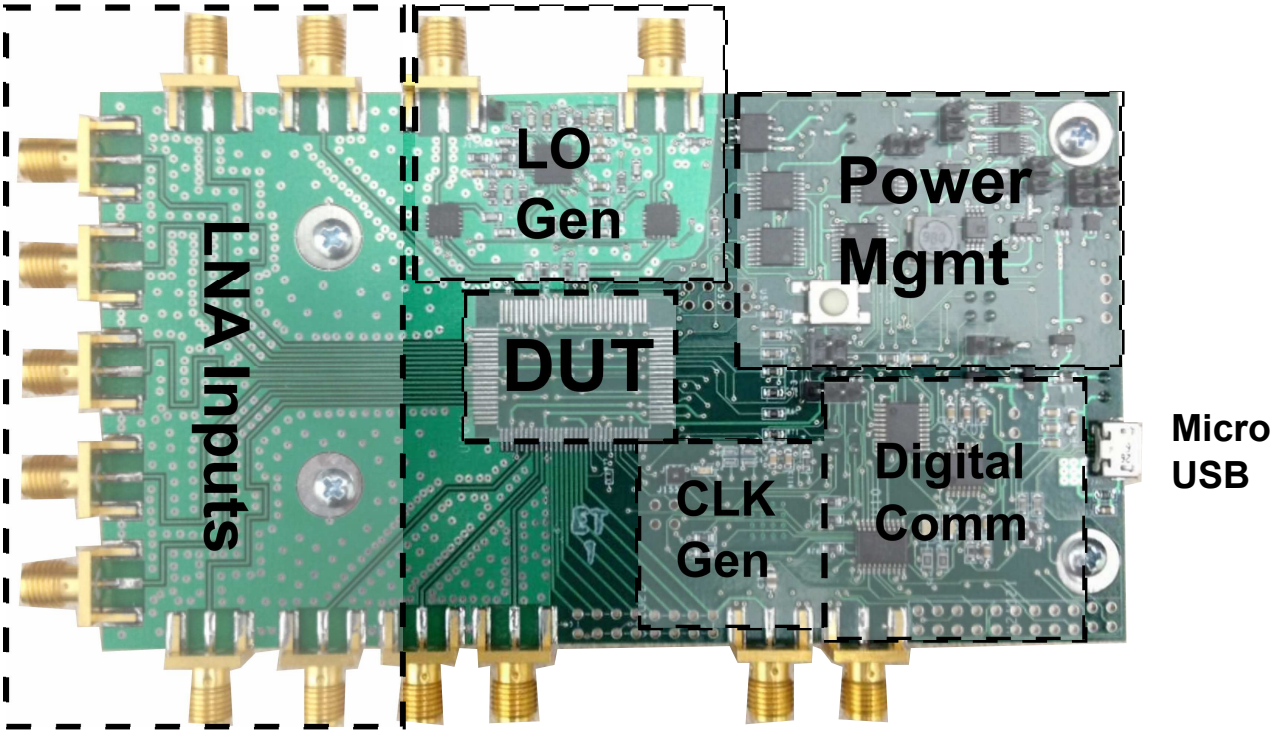


Figure 80: Picture of test PCB for RASP 3.0 RF chip.

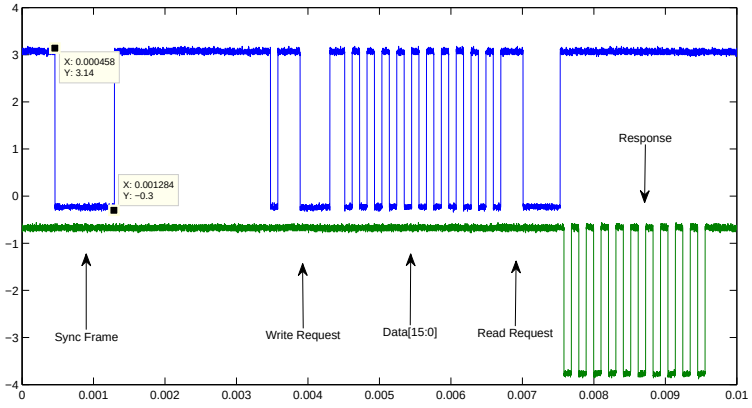
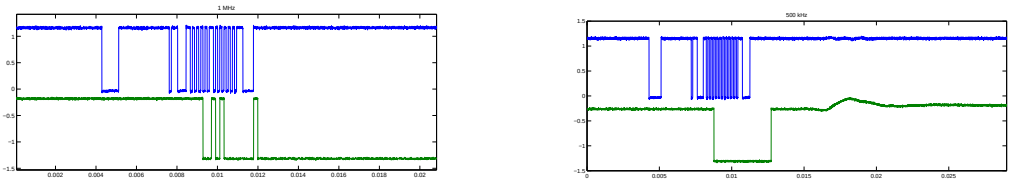
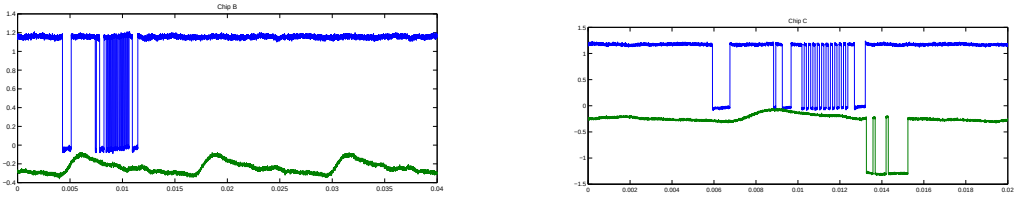


Figure 81: Writing and reading 0x5555 to register 7 on the RASP 3.0.



(a) (b)

Figure 82: (a) RASP 3.0RF chip A's response with a clock of 1 MHz (b) Chip A's response with a clock of 500 kHz.



(a) (b)

Figure 83: (a) RASP 3.0RF chip B always remained at a logic high level. (b) RASP 3.0RF chip C always responded with 0x4000's response with a clock of 1 MHz.

the same incorrect response, regardless of what data we attempted to write to the register. We show an example of this behavior in Fig. 7.5.2.

We tested more than 25 chips over a range of clock frequencies from 100 kHz to 70 MHz without any successful reads. Examples of responses from other chips (B and C) are shown in Fig. 7.5.2.

We reviewed the synthesis and routing scripts used to generate the openMSP430, and we discovered that parasitic extraction of the interconnect had not been run with the highest accuracy. As process nodes decrease, the delays caused by interconnect become a larger portion of the total delay in digital systems, so modeling them accurately is crucial. In the case of our synthesized system, during placement and routing we used capacitances generated by the LEF file, rather than using Cadence's field-based parasitic extraction tool, QRC.

We ran placement and routing routines to see how performing more accurate extraction would affect the system's performance. When timing analysis was run based on the first routing script, zero setup errors were reported. When timing analysis was run based on the second routing script, 186 setup errors were reported. There is a significant difference

between the accuracy of these two methods.

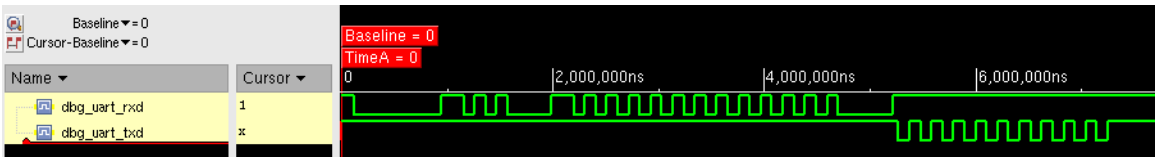
Next we investigated whether parasitics extracted by the more accurate solver were likely affecting the debug interface circuitry. We simulated a back-annotated version of the synthesized Verilog code and ran setup and hold timing checks. This was accomplished by exporting the timing results as an SDF file and then modifying the standard cell library such that a Verilog simulator could annotate the delays and run timing checks. Whenever a timing check failed, an unknown logic value x was asserted at the output of the flip flop. We constructed the simplest simulation possible – writing two bytes to a debug register and then reading them back out. An excerpt from both the modified standard cell library and the simulation testbench are given in Appendix C.

The debug interface was then simulated with and without timing checks. When timing checks were not run (but back-annotated delays were still used), the simulation indicated that the chip should work fine, as shown in Fig. 84(a). When timing checks were run, an unknown logic value was asserted upstream of the debug interface and propagated to the debug interface, causing a failure as shown in Fig. 84(b). The timing checks which caused errors were hold checks. This led us to attempt to run the chips at a lower Vdd (around 0.5-0.6V) and slower clock rates (100 kHz), but we still did not see a correct response from the debug interface.

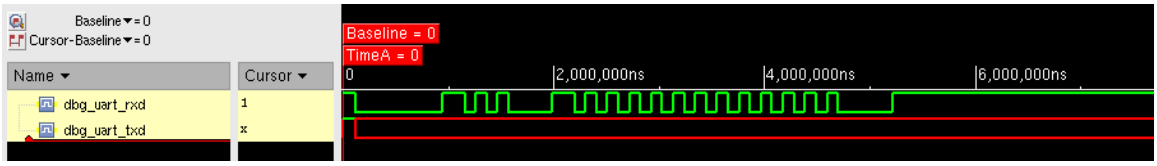
7.5.3 Recommendations for Future Designs

The primary mistake made in this design was neglecting to recognize the importance of interconnect parasitics at small technologies. We hadn't had these problems before since we'd been working in larger technologies where we could afford to not model interconnect parasitics as carefully. In future designs, CADSP members should be certain to run full QRC extraction of parasitics.

The best way to ensure that the correct procedure is being used is to follow what Cadence calls its "Foundation Flows." These consist of a set of TCL scripts and setup files that run through all the parasitic extraction, clock tree synthesis, timing optimization, and signoff steps. If at any point in the Foundation Flow a timing check is violated, the design should



(a)



(b)

Figure 84: (a) When setup and hold checks are not run in a simulation of the openMSP430, simulation predicts that the synthesized chip should behave as desired. After writing 0x5555 to a debug register, a read request returns 0x5555. (b) When setup and hold checks are run in the openMSP430 simulation, an unknown logic level is propagated from a failed check, and this eventually affects the chip's response to requests for debug communication.

be fixed. We should also be sure to run all verification simulations using the back-annotated Verilog files along with timing checks.

CHAPTER VIII

CONCLUDING REMARKS

This dissertation discussed both the mapping of circuits and systems onto FPAAs and the design of new FPAAs. Implementing systems on FPAAs informs the design of the next generation of chips. FPAAs allow for the quick implementation of analog signal processing systems, which are critical for low-power system design. The neuromorphic systems designed in this work may allow for useful neural algorithms (such as Independent Component Analysis) to be implemented in a low-power manner. We list the specific contributions of this dissertation in the following section.

8.1 Summary of Contributions

Chapter 2 introduced a transistor-only adaptive Winner-Take-All circuit. This circuit emulates a previous design [34] that used floating-gate transistors to model adaptation. Rather than use a floating-gate transistor, we model its tunneling and injection using two pFETs. This results in a circuit with dynamics that respond significantly upon a change in the step input but adapt to a less dramatic response on a long timescale. This behavior could be interpreted as a model of attention in neural systems.

Chapter 3 summarized both steady-state and dynamic experiments on the RASP 2.8a to show equivalence between CMOS dendrites and biological dendrites. It also discussed nonidealities in the model, due to either nonlinear effects or parasitic effects of the FPAA. Finally, we showed how a floating-gate dendrite could be compactly compiled in the routing fabric of an FPAA.

In Chapter 4, we implemented a subtractive synthesizer on the FPAA. We first constructed the building blocks of subtractive synthesis systems: a voltage-controlled oscillator, voltage-controlled filter, voltage-controlled amplifier, and an attack-decay-sustain-release envelope generator. We then made Simulink blocks for each of these subsystems and integrated automatic characterization into some of these blocks. Finally, we combined the

blocks to form a complete synthesizer.

Chapter 5 developed an automatic characterization routine for waveform shaping circuits on a neuromorphic system, the Neuron1D. We also performed learning experiments using the Spike-Timing Dependent Plasticity (STDP) synaptic learning rule. We demonstrated an in-system STDP curve. We then used STDP to strengthen coincident inputs into a neuron, even in the presence of significant noise. We also used STDP to strengthen the weight of the fastest-firing neuron in a group of neurons, which is analagous to a winner-take-all circuit.

In Chapter 6, we designed and laid out a Computational Analog Block (CAB) that creates current-mode DACs on the RASP 2.9v chip. We used these DACs as inputs to an analog image-processing system consisting of vector-matrix-multiplication circuits. This system found the edges of an image using the Sobel kernel, and it also blurred an image using a 9x9 smoothing kernel.

Chapter 7 discussed the design of the next generation of FPAAs, the RASP 3.0 family. We ported the floating-gate programmer from the RASP 2.9 to the RASP 3.0 infrastructure, which involved removing digital circuitry that was laid out by hand, adding inputs for a synthesized peripheral, reorganizing the logical structure of the programmer, and porting existing layout. In addition, we wrote a Verilog peripheral to interface the MSP430 and the programmer. We designed and laid out the RASP 3.0H chip, which includes a number of new features such as homeostatic synaptic scaling, STDP inhibitory synapses, and saddle-node neuron channels. We simulated a delay element used in the high-speed switch fabric of the RASP 3.0 RF. We designed a test PCB for the RF chip and tested the digital system. When the digital system did not behave as desired, we analyzed the placement and routing routines and found that the root cause was inadequate modeling of interconnect parasitics in the system.

PERIPHERAL REGISTERS OF THE RASP 3.0

Register [15:8]	15	14	13	12	11	10	9	8
Register [7:0]	7	6	5	4	3	2	1	0
gp_per_1 [15:8]	GP I/O							
gp_per_1 [7:0]	GP I/O [7:0]	sel avdd 2.5	select IVDD		sel avdd 6	sel vtun	clock scl	clock sda
gp_per_3 [15:8]	Z	ADC 1 sel input	loc dreset	12V disable	6V en	sel ESDVDD	IVDD sel	DVDD 2.5 sel
gp_per_3 [7:0]	bsa					bs sel	adc pwd	
Drain Control [15:8]	Drain DAC Code							timer onchip
Drain Control [7:0]	Drain Mux Control			use timer	use dac	i2V Mux Control		i2v circuit
Selected Gate [15:8]	Gate DAC Code							Z
Selected Gate [7:0]	Gate Mux Control			DVDD or INJVDD	Z			
Programmer ADC [15:8]	Z		Programmer ADC Code					
Programmer ADC [7:0]	Programmer ADC code							
Pulse Control [15:8]	pulse en	pulse rst	Exponent					Mantissa
Pulse Control [7:0]	Mantissa							
Row Select [15:8]	Row Select Value							
Row Select [7:0]	Row Select Value							
Col Select [15:8]	Column Select Value							
Col Select [7:0]	Column Select Value							
Prog Bits [15:8]	PROG	OV_N	MEAS_ADC	CLEAR_N_ADC	Z			
Prog Bits [7:0]	Z							
Tunnel Mux [15:8]	Tunnel Mux Value							
Tunnel Mux [7:0]	Tunnel Mux Value					ADC Shift		
Unselected pFET Gates [15:8]	Gate DAC Code							Z
Unselected pFET Gates [7:0]	Gate Mux Control			DVDD or INJVDD	Z			
Unselected nFET Gates [15:8]	Gate DAC Code							Z
Unselected nFET Gates [7:0]	Gate Mux Control			DVDD or INJVDD	Z			
Bias DACs [15:8]	Cascode DAC value							
Bias DACs [7:0]	VREF DAC Value							Cascode use DAC

APPENDIX B

RF PCB TEST PLATFORM SCHEMATICS

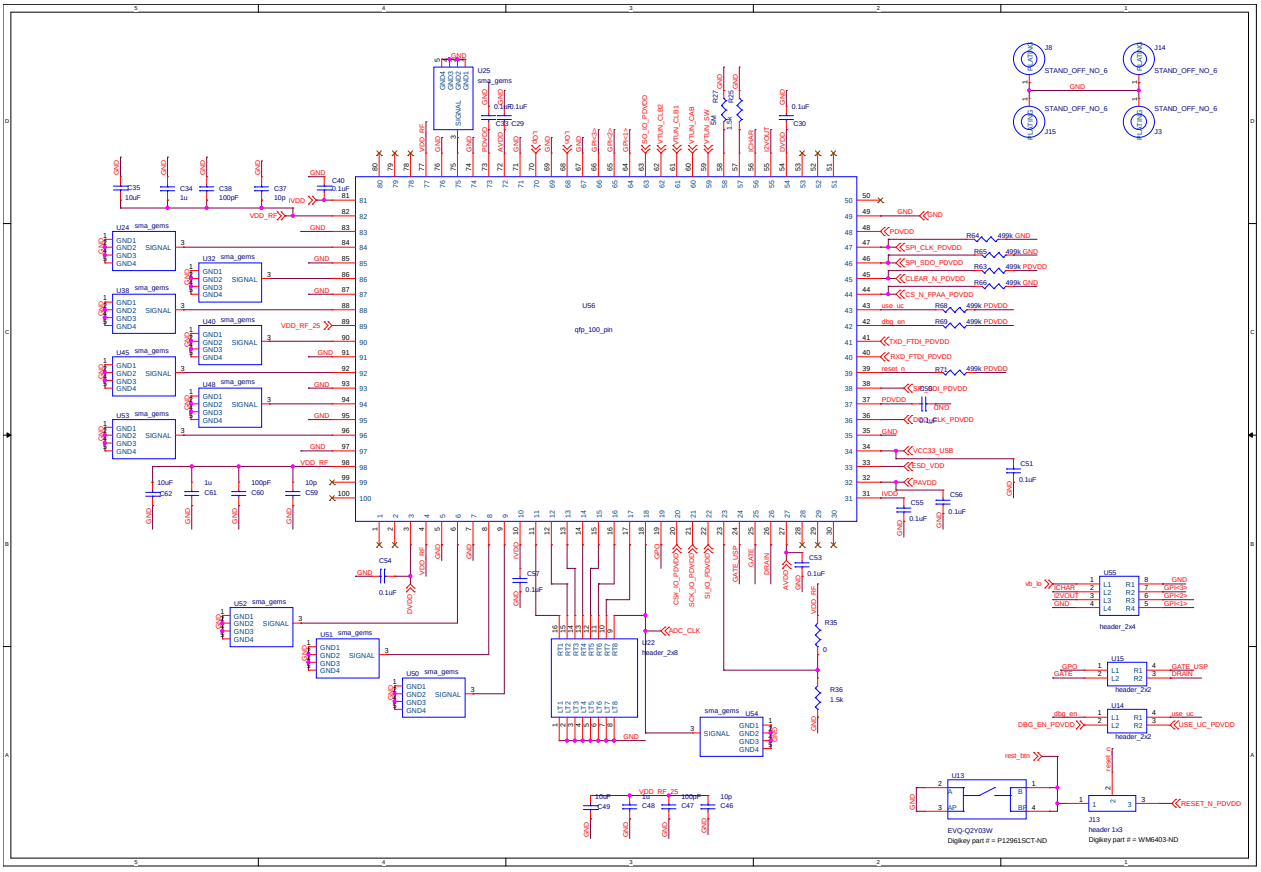


Figure 85: RASP portion of the RASP 3.0 RF PCB

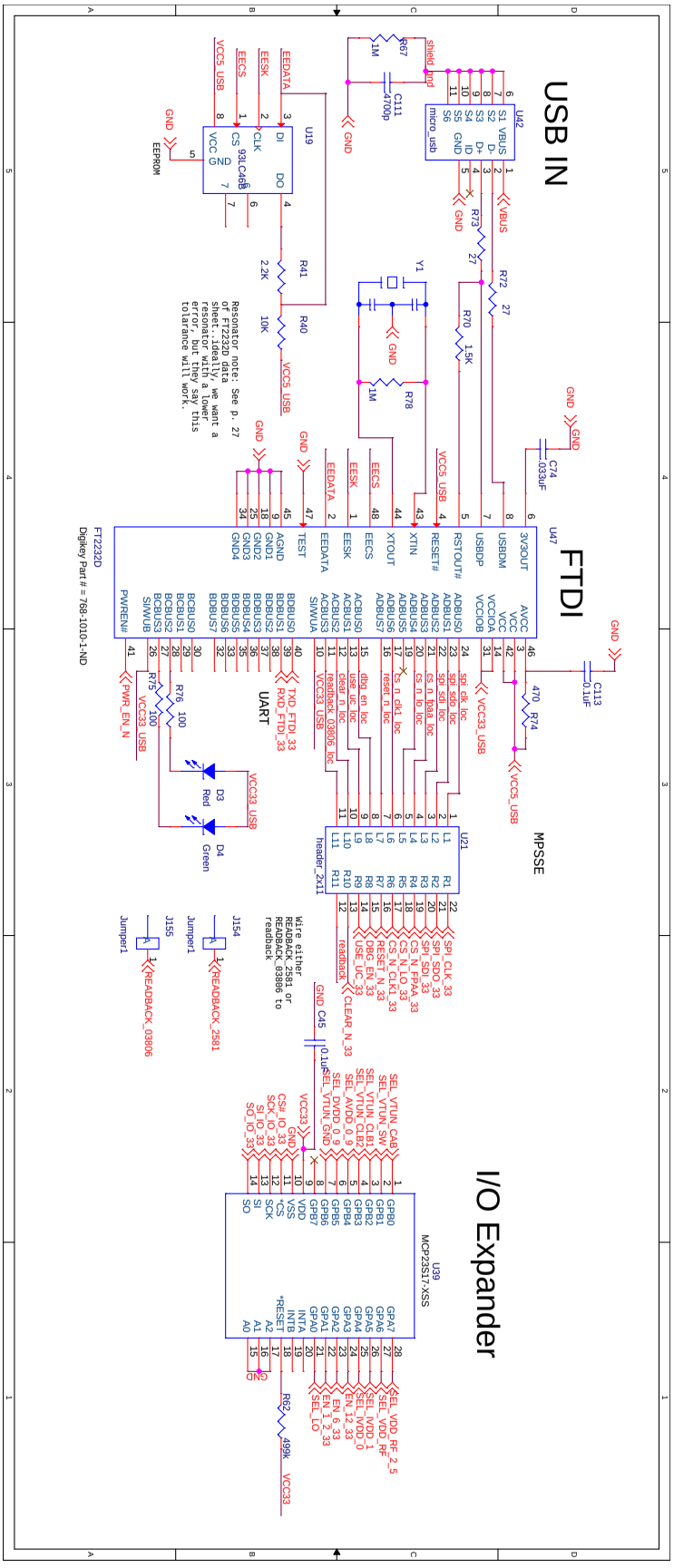


Figure 86: FTDI portion of the RASP 3.0 RF PCB

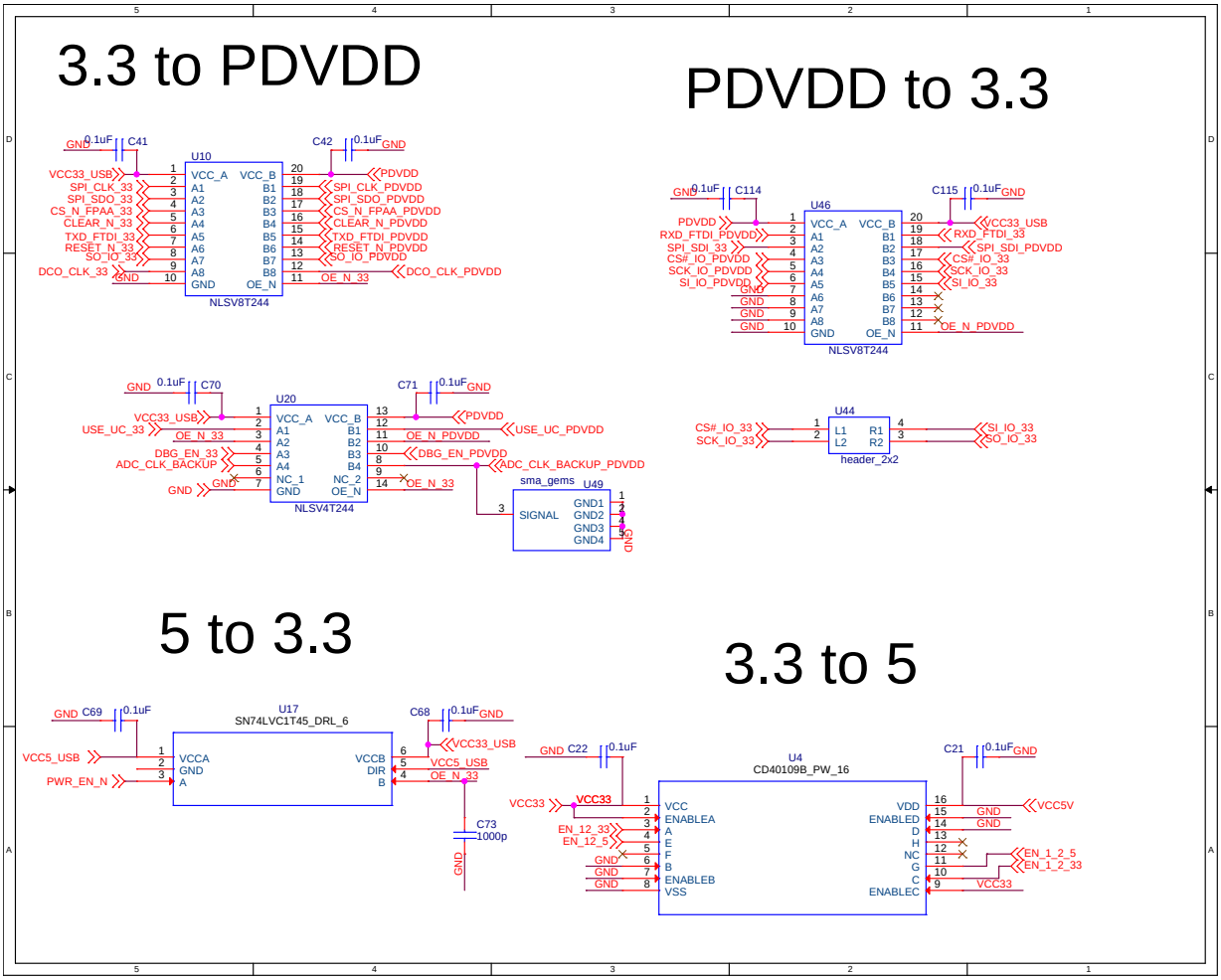


Figure 87: Level shifting portion of the RASP 3.0 RF PCB

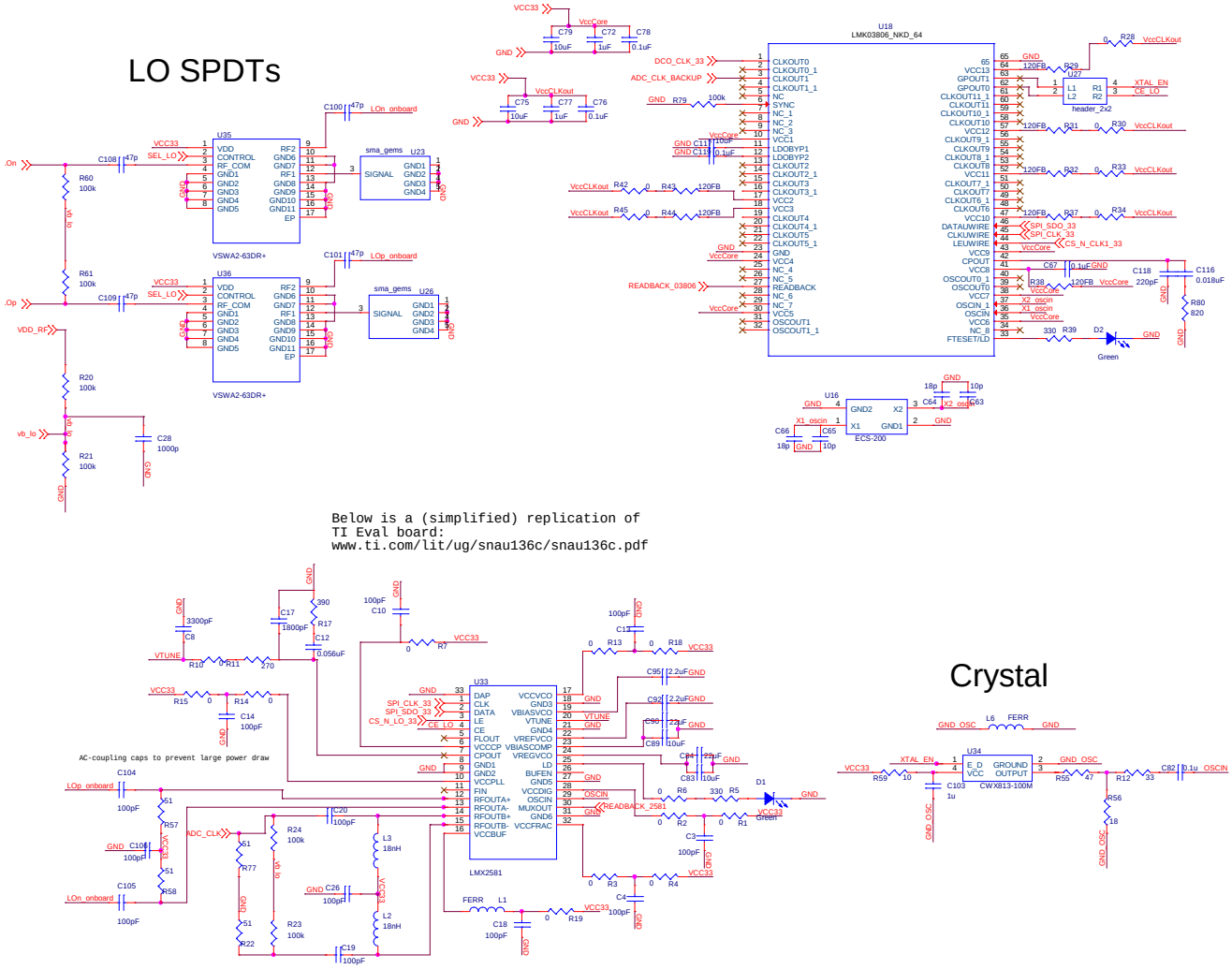


Figure 88: Clock portion of the RASP 3.0 RF PCB

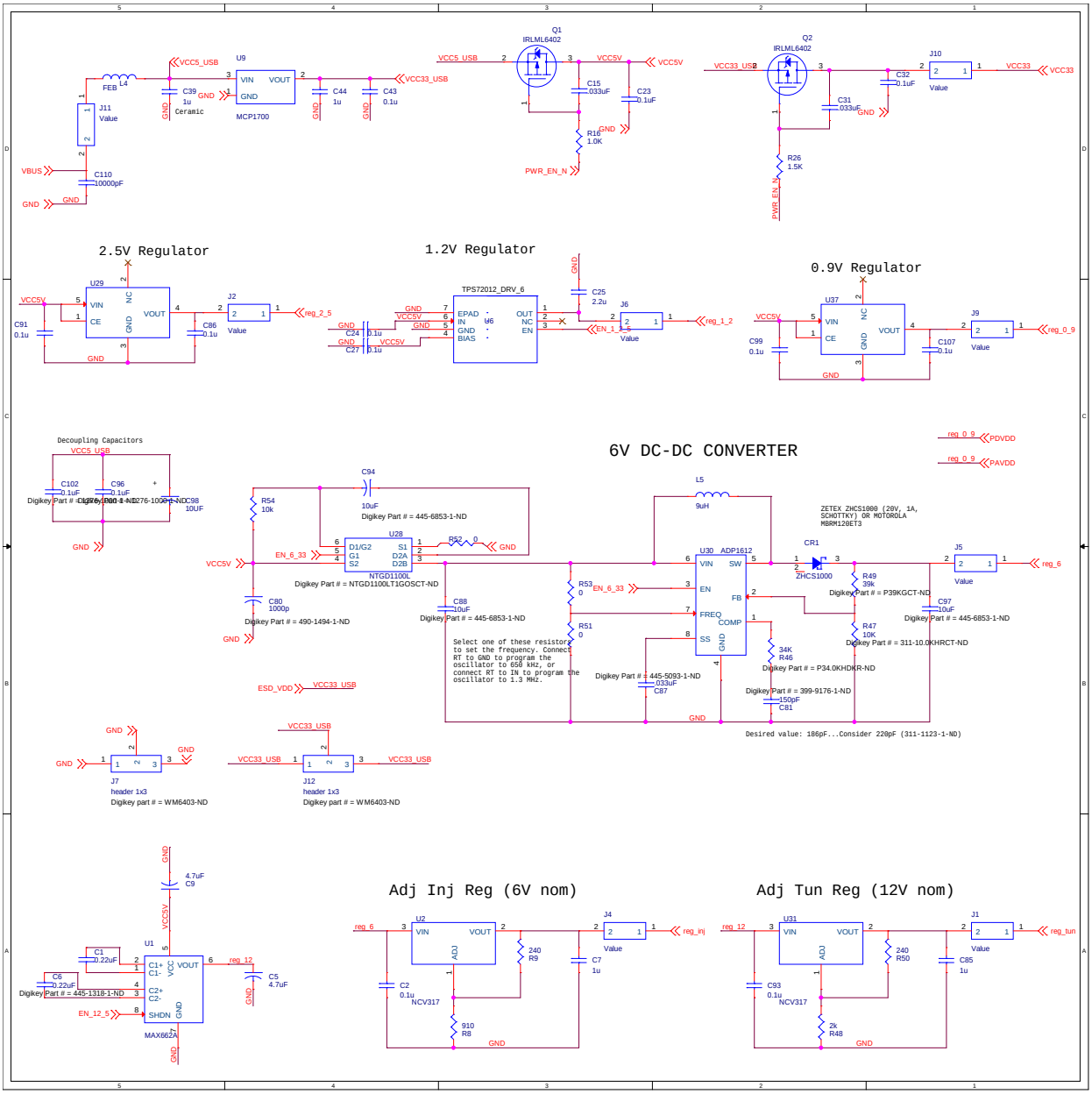


Figure 89: Power Regulation portion of the RASP 3.0 RF PCB

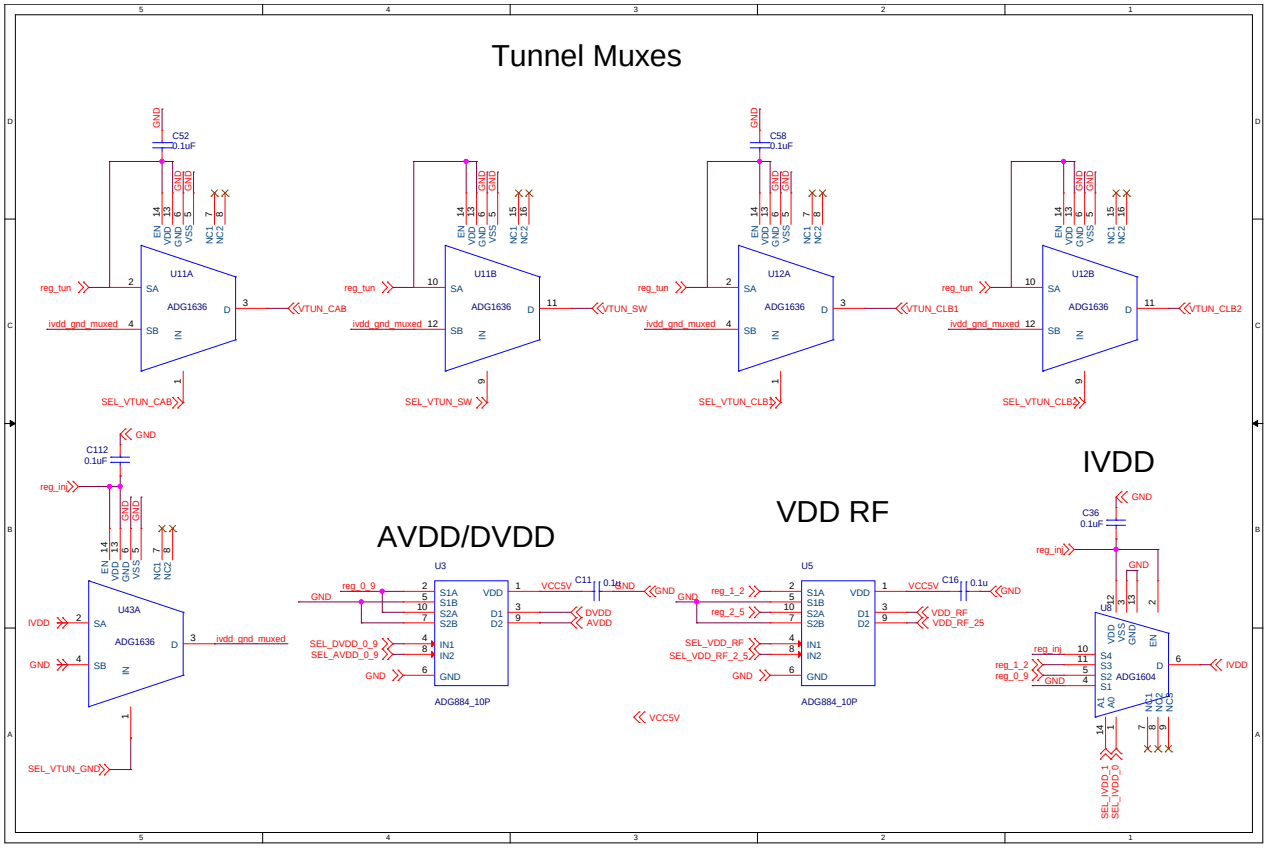


Figure 90: Power switching portion of the RASP 3.0 RF PCB

APPENDIX C

BACK-ANNOTATED AND TIMING-AWARE VERILOG SIMULATIONS OF THE RF CHIP

Below is an example of the method used to annotate the standard cell library for timing-aware, back-annotated verilog simulations:

```
module DFCNX1(CK,D,Q,QN,RN);
        reg notifier;
input CK,D,RN;
output Q,QN;
        buf(Q, ffout);
        not(QN,ffout);
        cadsp_dff (ffout, CK, D, RN, notifier);
        and(D_EQ_0_AN_RN_EQ_1, ~D, RN);
    specify
        (posedge CK => (Q +: D)) = (0, 0);
        (negedge CK => (Q +: D)) = (0, 0);
        (posedge CK => (QN -: D)) = (0, 0);
        (negedge CK => (QN -: D)) = (0, 0);
        (RN => Q) = (0, 0);
        (RN => QN) = (0, 0);
        $width(negedge RN &&& (D==0),0, notifier);
        $width(posedge CK,0, notifier);
        $width(negedge CK,0, notifier);
        $setuphold(posedge CK, D, 0, 0, notifier);
        $setuphold(posedge CK &&& (RN==1), posedge D, 0, 0, notifier);
        $setuphold(posedge CK &&& (RN==1), negedge D, 0, 0, notifier);
        $recrem(posedge RN, posedge CK &&& (D==1), 0, 0, notifier);
    endspecify
endmodule
```

```
primitive cadsp_dff(Q, CK, D, RN, notifier);
    output Q; reg Q;
    input CK, D, RN, notifier;
    table
    //  CK      D  RN notif   state     Q
    //---------------------------------
        (01)    0   1       ?   :   ?   :   0;  // posedge clk && D==0 => Q=0
        (01)    1   1       ?   :   ?   :   1;  // posedge clk && D==1 => Q=1
        (0?)    1   1       ?   :   1   :   1;  // rising edge with clk = x or z => maintain state
        (0?)    0   1       ?   :   0   :   0;
        (?0)    ?   ?       ?   :   ?   :   -;
        ?       *   ?       ?   :   ?   :   -;
        ?       ?   0       ?   :   ?   :   0;
        ?       ?   (01)    ?   :   ?   :   -;
        ?       ?   (10)    ?   :   ?   :   0;
        ?       ?   ?       *   :   ?   :   x;
    endtable
endprimitive
```

Below is a condensed version of the testbench used to test for functionality of the debug interface:

```
‘timescale 1ns / 1ps
module tb_openMSP430;
\\INPUTS (inputs defined as registers here)
\\OUTPUTS (outputs defined as wires here)

openMSP430 dut( connections between inputs/outputs and DUT)

always
    dco_clk = !dco_clk

initial
begin
    $shm_open("./WAVEFORM");
    $shm_probe(tb_openMSP430, "AS");
```

```
    $sdf_annotate("openMSP430_setup_split.sdf",tb_openMSP430.dut,,"sdffile.log");

    // Insert inital values for registers

    // reset device

    #1000 reset_n = 0;
    #10000 reset_n = 1;

    #1 tx_byte(8'h80); // init
    #1 tx_byte(8'h85); // write
    #1 tx_byte(8'h55); // data 1
    #1 tx_byte(8'h55); // data 2
    #1 tx_byte(8'h05); // read
    #5000000

    #1 $finish;
end

task tx_byte;
    input [7:0] data;
    begin
        // 8680 for 115K baud rate at timescale = 1ns
        // 104160 for 9.6K baud rate at timescale = 1ns
        // 16666 for 600 baud rate at timescale = 100ns
        #104160 dbg_uart_rxd = 0; // start bit
        for (i=0; i<8; i=i+1) begin
            #104160 dbg_uart_rxd = data[i]; // LSB first
        end
        #104160 dbg_uart_rxd = 1; // stop bit
    end
endtask

endmodule
```

APPENDIX D

TECHNICAL ACKNOWLEDGEMENTS

Thanks to Mitch Levine for his initial efforts debugging the adaptive WTA circuit.

REFERENCES

[1] Allen, C. and Stevens, C. F., "An evaluation of causes for unreliability of synaptic transmission," *Proceedings of the National Academy of Sciences*, vol. 91, no. 22, pp. 10380–10383, 1994.

[2] Apple, "Logic express 9 appendix a, section 3: How subtractive synthesizers work."

[3] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C., and Hasler, P., "A floating-gate-based field programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[4] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C., and Hasler, P., "A floating-gate-based field-programmable analog array," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 9, pp. 1781–1794, 2010.

[5] Basu, A. and Hasler, P. E., "Nullcline-based design of a silicon neuron," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 11, pp. 2938–2947, 2010.

[6] Basu, A. and Hasler, P. E., "A fully integrated architecture for fast and accurate programming of floating gates over six decades of current," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 6, pp. 953–962, 2011.

[7] Basu, A., Petre, C., and Hasler, P., "Bifurcations in a silicon neuron," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 428–431, IEEE, 2008.

[8] Bi, G. and Poo, M., "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *The Journal of Neuroscience*, vol. 18, no. 24, pp. 10464–10472, 1998.

[9] Branco, T., Clark, B. A., and Hausser, M., "Dendritic discrimination of temporal input sequences in cortical neurons," *Science*, vol. 329, pp. 1671–1675, September 2010.

[10] Brink, S., Nease, S., and Hasler, P., "Computing with networks of spiking neurons on a biophysically motivated floating-gate based neuromorphic integrated circuits," *Neural Networks*, Accepted for Publication.

[11] Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., and Degnan, B., "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *Biomedical Circuits and Systems, IEEE Transactions on*, Accepted for Publication.

[12] Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., and Degnan, B., "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," 2012.

[13] Brink, S., Ramakrishnan, S., Nease, S., Shapero, S., Wunderlich, R., Hasler, P., Basu, A., and Degnan, B., "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems (To be Submitted for Publication)*.

[14] Caporale, N. and Dan, Y., "Spike timing-dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[15] Destexhe, A., "Dendrites do it in sequences," *Science*, vol. 329, pp. 1611–1612, September 2010.

[16] Elias, J. G., "Artificial dendritic trees," *Neural Computation*, vol. 5, pp. 648 – 663, 1993.

[17] Farquhar, E. and Hasler, P., "A bio-physically inspired silicon neuron," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, pp. 477 – 488, 2005.

[18] Földiak, P., "Forming sparse representations by local anti-hebbian learning," *Biological cybernetics*, vol. 64, no. 2, pp. 165–170, 1990.

[19] Girard, O., "Openmsp430 project," *available at opencore. org*, 2010.

[20] Gordon, C., Farquhar, E., and Hasler, P., "A family of floating-gate adapting synapses based upon transistor channel models," *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04.*, vol. 1, pp. 317 – 320, 2004.

[21] Haas, J. S., Nowotny, T., and Abarbanel, H., "Spike-timing-dependent plasticity of inhibitory synapses in the entorhinal cortex," *Journal of Neurophysiology*, vol. 96, pp. 3305–3313, 2006.

[22] Hall, T. S., Twigg, C. M., Gray, J. D., Hasler, P., and Anderson, D. V., "Large-scale field-programmable analog arrays for analog signal processing," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 11, pp. 2298–2307, 2005.

[23] Hasler, P., Diorio, C., Minch, B., and Mead, C., "Single transistor learning synapses," *Advances in neural information processing systems*, pp. 817–826, 1995.

[24] Hasler, P., Kucic, M., and Minch, B. A., "A transistor-only circuit model of the autozeroing floating-gate amplifier," in *42nd Midwest Symposium on Circuits and Systems*, (Las Cruces, NM), pp. 157–160, Aug. 1999.

[25] Hasler, P., Koziol, S., Farquhar, E., and Basu, A., "Transistor channel dendrites implementing hmm classifiers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 3359 – 3362, 2007.

[26] Hasler, P., Minch, B. A., and Diorio, C., "Adaptive circuits using pfet floating-gate devices," in *Advanced Research in VLSI, 1999. Proceedings. 20th Anniversary Conference on*, pp. 215–229, IEEE, 1999.

[27] Hebb, D. O., *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[28] Hille, B., *Ion Channels of Excitable Membranes, Third Edition*. Sunderland, MA: Sinauer Associates, Inc, 2001.

[29] Huovilainen, A. and Välimäki, V., "New approaches to digital subtractive synthesis," in *Proceedings of the International Computer Music Conference, Barcelona, Spain*, pp. 399–402, 2005.

[30] Kepecs, A., van Rossum, M. C., Song, S., and Tegner, J., "Spike-timing-dependent plasticity: common themes and divergent vistas," *Biological cybernetics*, vol. 87, no. 5-6, pp. 446–458, 2002.

[31] Koch, C., *Biophysics of Computation*. New York, NY: Oxford University Press, 1999.

[32] Koch, C. and Segev, I., "The role of single neurons in information processing," *Nature Neuroscience*, vol. 3, pp. 1171–1177, Nov. 2000.

[33] Koziol, S., Schlottmann, C., Basu, A., Brink, S., Petre, C., Degnan, B., Ramakrishnan, S., Hasler, P., and Balavoine, A., "Hardware and software infrastructure for a family of floating-gate based fpaas," *Circuits and Systems, 2010. ISCAS 2010. IEEE International Symposium on*, pp. 2794 – 2797, 2010.

[34] Kruger, W. F., Hasler, P., Minch, B., and Koch, C., "An adaptive wta using floating gate technology," in *Proc. Advances in Neural Information Processing Systems Conference 1996*, (Denver, CO, USA), pp. 720–726, Dec. 1996.

[35] Lazzaro, J., Ryckebusch, S., Mahowald, M. A., and Mead, C. A., "Winner-take-all networks of o(n) complexity," in *NIPS 1*, (Denver, CO), pp. 703–711, 1989.

[36] Lee, E. K. F. and Hui, W. L., "A novel switched-capacitor based field-programmable analog array architecture," *Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1-2, pp. 35–50, 1998.

[37] Liu, S.-C., Kramer, J., Indiveri, G., Delbruck, T., and Douglas, R., *Analog VLSI: Circuits and Principles*. Cambridge, MA: The MIT Press, 2002.

[38] Liu, S.-C. and Mockel, R., "Temporally learning floating-gate vlsi synapses," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 2154–2157, IEEE, 2008.

[39] Liu, S.-C. and Oster, M., "Feature competition in a spike-based winner-take-all vlsi network," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4–pp, IEEE, 2006.

[40] London, M. and Hausser, M., "Dendritic computation," *Annual Review of Neuroscience*, vol. 28, pp. 503–532, July 2005.

[41] Maass, W., "On the computational power of winner-take-all," *Neural Computation*, vol. 12, no. 11, pp. 2519–2535, 2000.

[42] Masquelier, T., Guyonneau, R., and Thorpe, S. J., "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PLoS one*, vol. 3, no. 1, p. e1377, 2008.

[43] Mead, C., *Analog VLSI and neural systems*. Addison-Wesley, 1989.

[44] Mead, C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[45] Moog, R., "Electronic, high-pass, and low-pass filters employing the base to emitter diode resistance of bipolar transistors," 1969.

[46] Nease, S., Brink, S., and Hasler, J., "Stdp-enabled learning on a reconfigurable neuromorphic platform," in *Circuit Theory and Design (ECCTD), 2013 21st European Conference On*, 2013.

[47] Ramakrishnan, S., Hasler, P., and Gordon, C., "Floating gate synapses with spike-time-dependent plasticity," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 5, no. 3, pp. 244–252, 2011.

[48] Ramakrishnan, S., Hasler, P., and Gordon, C., "Floating gate synapses with spike-time-dependent plasticity," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, pp. 244 – 252, 2011.

[49] Rasche, C. and Douglas, R. J., "Forward- and backpropagation in a silicon dendrite," *IEEE Transactions on Neural Networks*, vol. 12, pp. 386 – 393, Mar. 2001.

[50] Sarpeshkar, R., "Analog versus digital: extrapolating from electronics to neurobiology," *Neural computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[51] Schlottmann, C., Petre, C., and Hasler, P., "A high-level simulink-based tool for fpaa configuration," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, no. 99, pp. 10–18, 2010.

[52] Schlottmann, C., Petre, C., and Hasler, P., "Vector matrix multiplier on field programmable analog array," *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 1522 – 1525, 2010.

[53] Schlottmann, C., Shapero, S., Nease, S., and Hasler, P., "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *Journal of Solid-State Circuits (Submitted for Publication)*.

[54] Schlottmann, C. R., "A coordinated approach to reconfigurable analog signal processing." Georgia Tech Dissertation, July 2012.

[55] Segev, I., Rinzel, J., and Shepherd, G. M., *The Theoretical Foundation of Dendritic Function: Selected Papers of Wilfrid Rall with Commentaries*. Cambridge, MA: The MIT Press, 1995.

[56] Shapero, S. and Hasler, P., "Precise programming and mismatch compensation for low power analog computation on an fpaa," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems, submitted for review*, 2012.

[57] Stilson, T. and Smith, J., "Analyzing the moog vcf with considerations for digital implementation," in *Proceedings of the International Computer Music Conference, Hong Kong*, 1996.

[58] Stinchombe, T. E., "Analysis of the moog transistor ladder and derivative filters," 2008.

[59] Twigg, C., Gray, J., and Hasler, P., "Programmable floating gate fpaa switches are not dead weight," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 169–172, May 2007.

[60] Wang, Y. and Liu, S.-C., "Input evoked nonlinearities in silicon dendritic circuits," *IEEE International Symposium on Circuits and Systems*, pp. 2894 – 2897, 2009.

[61] Williams, R., "Triangle to sine conversion with otas."

[62] Wunderlich, R. B., Adil, F., and Hasler, P., "Floating gate-based field programmable mixed-signal array," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 8, pp. 1496–1505, 2013.

[63] Ying, G., Kuehlmann, A., Kundert, K., Gielen, G., Grimme, E., O'Leary, M., Tare, S., and Wong, W., "Guess, solder, measure, repeat-how do i get my mixed-signal chip right?," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*, pp. 520–521, IEEE, 2009.

VITA

Stephen Nease was born and raised in Memphis, TN, where he graduated from Memphis University School in 2005. He received the B.S. degree (summa cum laude) in electrical engineering from the Rose-Hulman Institute of Technology in 2009. Stephen's senior project was the digitization of a sonic well depth finder. During his undergrauate studies, Stephen interned at Pacer Digital Systems, Inc., where he built and tested control systems for cryogenic freezers. He also interned at Boston Scientific, Cardiac Rhythm Management, where he wrote software to evaluate integrated circuit tests.

His research interests include neuromorphic engineering, analog signal processing, and reconfigurable mixed-signal systems.